Reconstruction de primitives géométriques avec de la lumière non structurée

par

Julien Prémont

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, mai 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-83701-6

Our file Notre référence
ISBN: 978-0-494-83701-6

Le 24 mai 2011

le jury a accepté le mémoire de Monsieur Julien Prémont
dans sa version finale.

Membres du jury

Professeur Pierre-Marc Jodoin
Directeur de recherche
Département d'informatique

Monsieur Marc-Antoine Drouin
Membre externe
Conseil national de recherches Canada

Professeur Bernard Colin
Président rapporteur
Département de mathématiques

SOMMAIRE

La vision 3D est l'art de remplacer le système de vision humain par un ordinateur. Si des images peuvent aisément être acquises par des caméras, l'analyse de la profondeur et de la forme des objets est une tâche autrement plus complexe. La reconstruction d'un objet en 3D se divise généralement en deux étapes : l'estimation de la profondeur de points discrets de la scène (qui correspondent généralement aux pixels d'une image), puis le traitement de ces points, qui peut prendre différentes formes selon le type d'application. Plusieurs techniques existent pour estimer la profondeur de points, chacune fonctionnant bien sous certaines conditions favorables. Parmi ces conditions, on retrouve souvent l'immobilité des objets à reconstruire et la constance de l'illumination (ou sinon, une structure dans ses variations – la lumière structurée). Ce mémoire présente une technique qui non seulement permet, mais requiert et se base sur de grandes variations d'illumination, comme celles causées par la projection d'une séquence vidéo sur l'objet. Cette technique, que nous avons nommée lumière non structurée (LNS), se base sur la cooccurrence statistique d'événements dans deux séquences vidéo : celle projetée et celle filmée par la caméra.

Par la suite, nous utilisons une variante de l'algorithme d'estimation RANSAC pour diviser les points 3D en plusieurs primitives géométriques et estimer la nature de ces primitives (plans, sphères, cylindres, etc.).

REMERCIEMENTS

Je veux remercier M. Pierre-Marc Jodoin, mon directeur de recherche, pour sa patience et sa confiance, de même que pour le financement qu'il m'a fourni.

Je tiens ensuite à remercier M. Marc-Antoine Drouin pour ses précieux conseils et enseignements, sa patience infinie et ses encouragements constants. Je remercie également Marc-Antoine et le Conseil National de Recherche du Canada pour leur accueil à Ottawa, l'opportunité de travailler sur ce projet et pour les résultats qu'ils m'ont fournis et qui figurent dans ce mémoire.

Je remercie également le troisième membre du jury, M. Bernard Colin.

J'adresse des remerciements tout particuliers à MM. Christophe Rosenberger et Pierre-Marc Jodoin pour m'avoir permis d'aller poursuivre mes travaux à Caen pendant quatre mois.

Je remercie les étudiants (et professeurs) du MoIVRe, de même que mes voisins d'étage aux résidences (à Sherbrooke et à Caen), mes parents, ma soeur, mes amis de Québec, qui ont tous contribué (et nui, fort heureusement) au bon avancement de mes travaux. Sans eux, ma maîtrise n'aurait pas été la partie de plaisir qu'elle a été.

Je remercie enfin mon père pour la révision orthographique de ce mémoire.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

D'*Avatar* à *Harry Potter*, en passant par la plus récente Coupe du monde de soccer et un concert de U2, le cinéma et les diffusions d'événements en 3D font couler beaucoup d'encre depuis quelques années. Ces applications ludiques sont d'ailleurs maintenant indissociablement associées au concept de la vision 3D dans l'imaginaire collectif. Pourtant, la vision 3D est en réalité le concept inverse, c'est-à-dire l'utilisation de l'informatique pour analyser des images 2D afin de reconstituer une scène (inconnue) en 3D, plutôt que pour créer des images 2D représentant fidèlement une scène (connue) en 3D. Le domaine de la vision 3D présente des applications très intéressantes et utiles. Des techniques ont été développées pour effectuer des tâches aussi variées que la création automatique de cartes topographiques à partir d'images satellites [1], la numérisation d'objets en temps réel [27] ou l'analyse du comportement humain dans des lieux publics [4].

Les défis dans la reconstitution de scènes ou d'objets en 3D sont multiples. Le processus pour y arriver peut être divisé en deux étapes : (1) estimer la position de points 3D de la scène et (2) donner un sens à cette information d'une façon qui est propre à chaque application. Par exemple, il est possible de créer un maillage à partir de points 3D résultant d'une reconstruction, ou encore, dans le cas du contrôle de qualité, vérifier que les mesures correspondent aux normes établies.

Pour estimer la profondeur des objets, de nombreux indices peuvent être utilisés. Par exemple, le flou [13], la variation des couleurs [31], le mouvement [10] ou la disparité

binoculaire [45] sont souvent utilisés. Ce dernier indice est utilisé dans le cadre de la stéréovision, l'utilisation de deux images prises de points de vue légèrement différents[1] (nommées image de gauche et de droite), simulant en quelque sorte le système visuel humain. La disparité binoculaire est alors la distance entre la représentation d'un même point 3D dans les deux images. Avec cette disparité, la profondeur se calcule directement en vertu de simples règles géométriques. Par contre, l'art d'identifier les deux pixels correspondant à un même point de la scène n'est pas simple. De nombreuses techniques ont été proposées pour faciliter cette mise en correspondance de pixels. Entre autres, il est possible d'identifier uniquement chaque point de la scène en y projetant des motifs de lumière spécialement conçus. De tels systèmes utilisent un projecteur et une ou plusieurs caméras. Cette technique, nommée lumière structurée, n'est toutefois pas toujours appropriée, notamment lorsqu'une autre projection est faite sur la scène. La méthode de reconstruction 3D que nous proposons, nommée *LNS*[2], utilise la projection d'une séquence vidéo, quelle qu'elle soit. Se basant sur le mouvement des objets dans celle-ci, LNS utilise la cooccurrence statistique d'événements [4] dans les images de gauche et de droite pour mettre en correspondance leurs pixels et estimer la profondeur des points de la scène. Le chapitre 2 est consacré à la description et à l'étude de notre méthode de mise en correspondance de pixels, dite par lumière non structurée.

Les points 3D obtenus présentent toutefois peu d'intérêt lorsque considérés individuellement. Nous présentons donc au chapitre 3 une façon de séparer ces points en groupes appartenant aux différents objets de la scène. Nous utilisons pour ce faire une variante de l'algorithme stochastique RANSAC. Nous étudions également d'autres algorithmes pour segmenter les points 3D en surfaces planes.

Nous offrons enfin une conclusion et proposons des améliorations ou possibilités de développements futurs (chapitre 5). Dans un premier temps, le chapitre 1 décrit le type

1. Les points de vue doivent différer par au moins une translation. Modifier l'orientation de la caméra n'est pas suffisant.
2. LNS : Lumière non structurée

de systèmes sur lequel se base LNS et présente quelques applications qui ont motivé nos travaux. La section 2.1 donne une vue d'ensemble du fonctionnement de LNS.

CHAPITRE 1

Description du système et de ses applications

1.1 Applications

Lors d'une projection vidéo, la surface servant d'écran doit idéalement être plane et parallèle au plan image du projecteur. Autrement, la nature de la surface de projection induit une distorsion dans les images vues par l'utilisateur. Elles n'apparaissent alors pas rectangulaires comme elles le devraient. Notre méthode permet, à l'aide d'une caméra, d'utiliser les images projetées (et vues par la caméra) afin d'analyser la géométrie de la surface de projection, pour ensuite appliquer une transformation aux images à projeter qui élimine la distorsion causée par la surface de projection. La figure 1.2 présente un exemple de correction de distorsion réalisée par notre méthode.

La méthode que nous proposons ici est utile dans des applications où une séquence vidéo est projetée sur une surface potentiellement irrégulière et susceptible de se déplacer pendant la projection. On rencontre ce type d'applications principalement dans le domaine

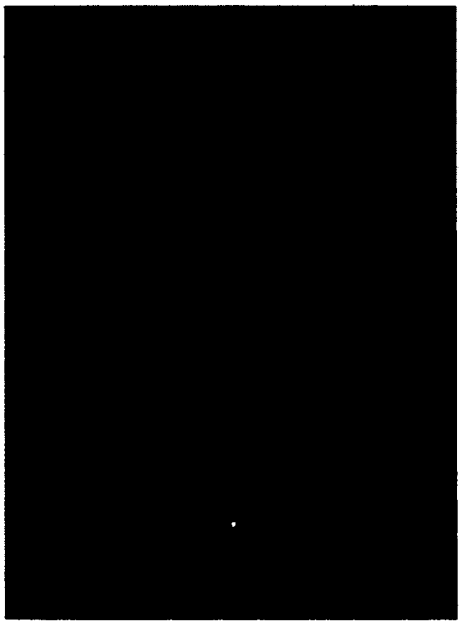


Figure 1.1 – Une projection est faite sur l'église Saint-Roch, à Québec, dans le cadre du spectacle *Fantasmagorie*, en décembre 2010.

des arts, par exemple dans des spectacles à grand déploiement ou dans des présentations interactives. Dans les dernières années, plusieurs spectacles, dont le Moulin à images et la cérémonie d'ouverture des Jeux olympiques de Vancouver, ont utilisé des projections vidéo. Les projections architecturales comme *Fantasmagorie*[1], sur l'église Saint-Roch à Québec en décembre 2010 (voir figure 1.1), ou celles conçues par l'artiste Mr. Beam [2], dans le cadre de différents événements au Bénélux, sont d'autres exemples de spectacles utilisant des projections.

Dans des proportions plus modestes, on peut également penser à des appareils photo numériques dotés d'un projecteur. De tels appareils existent déjà et il est permis de penser que leur nombre ne fera qu'augmenter dans les prochaines années. Notre méthode rendrait alors possible la projection de films ou séquences vidéo maison sans distorsion géométrique, et ce sur des surfaces en tout genre.

1. Fantasmagorie, une création de BlackOut Design.

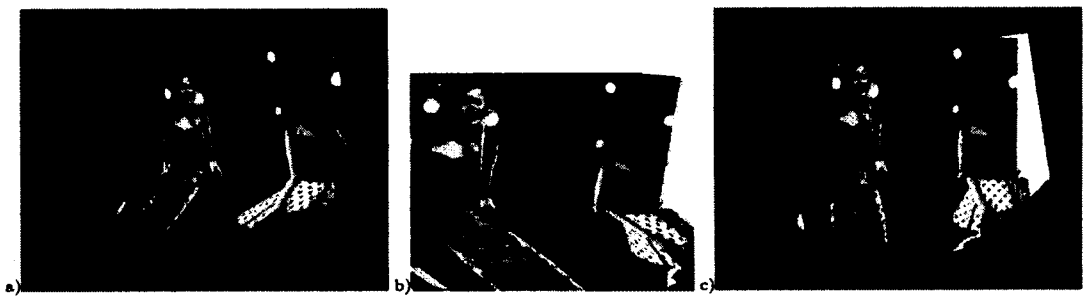


Figure 1.2 – (a) L'image projetée est déformée par la géométrie de la surface de projection. En appliquant une transformation aux images avant de les projeter (b), notre méthode permet d'éliminer la distorsion induite par la surface (c).

1.2 Système type

Notre méthode, LNS, peut fonctionner avec toute application utilisant une caméra et un projecteur. Il est toutefois nécessaire que la position relative des deux appareils ne change pas au cours d'une projection. Plus spécifiquement, l'intérêt principal de notre méthode réside dans les systèmes où la position relative de la scène et du projecteur est susceptible de changer pendant la projection. Notre méthode permet alors d'adapter dynamiquement la correction de distorsion géométrique appliquée à la séquence projetée. La figure 1.2 présente une configuration type d'un système utilisant notre méthode.

1.3 Travaux existants

La principale problématique d'un tel système réside dans la mise en correspondance de pixels de la caméra avec ceux du projecteur. Notons d'abord que nous traitons le projecteur comme une seconde caméra. Ainsi, les nombreuses approches de mise en cor-

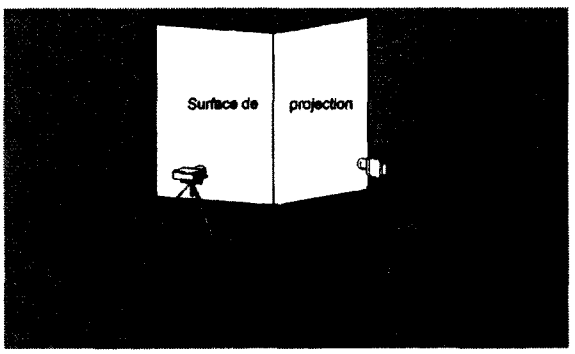


Figure 1.3 – Système type pour notre méthode, composé d'une caméra, d'un projecteur et d'une scène non monoplane.

respondance proposées pour deux caméras pourraient être utilisées dans notre système. Or, en raison des différentes contraintes inhérentes aux applications de notre méthode, aucune des approches classiques ne peut être utilisée avec succès dans un système comme le nôtre. Les plus importantes approches de la littérature sont décrites ici.

1.3.1 Stéréovision

La méthode classique de stéréovision vise à faire correspondre les pixels de deux caméras, en se basant sur leur intensité (ou couleur). La mise en correspondance se fait au moyen d'une fonction de coût comparant un pixel p_g de l'image de gauche et son voisinage aux pixels p_d de l'image de droite et leur voisinage respectif. Une des fonctions de coût les plus populaires est la somme des carrés des différences entre les intensités des pixels : $C(p_g, p_d) = \sum_{p_{a_i} \in \Gamma(p_a)} (p_{g_i} - p_{d_i})^2$, où a est soit g ou d, et $\Gamma(p)$ un voisinage local du pixel p. Les correspondances sont alors établies selon une méthode d'optimisation, soit vorace ou globale, trouvant les correspondances de pixels pour lesquelles le coût sera minimal. Les méthodes voraces sont rapides, mais souffrent grandement du manque de texture. Les méthodes globales (comme le recuit simulé et la coupe de graphe), en revanche, donnent de meilleurs résultats, mais sont considérablement plus lentes. Une étude comparative

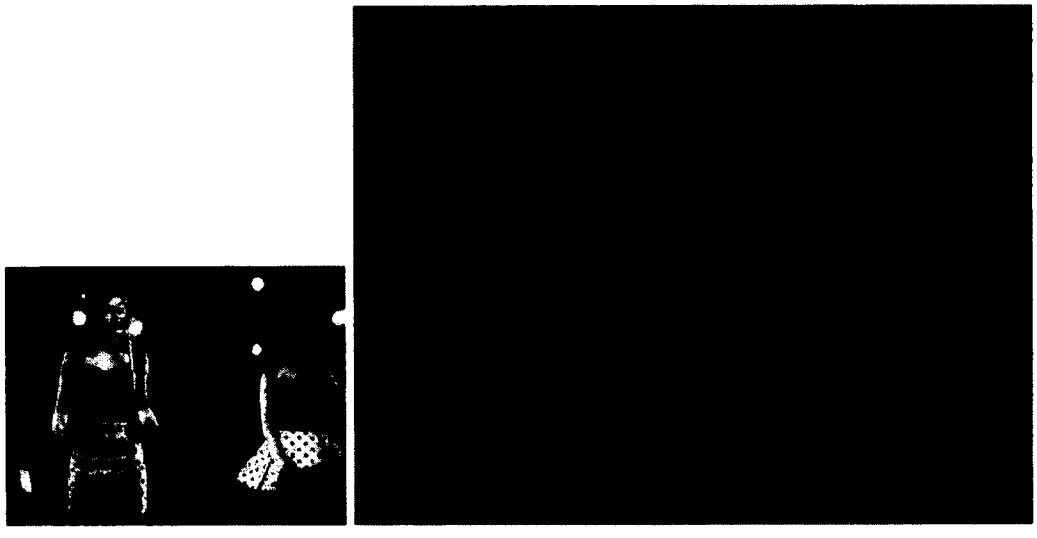


Figure 1.4 – L'image de la caméra (droite) a subi une forte dégradation de contraste en raison du court temps d'exposition. Elle n'a plus les mêmes couleurs que l'image projetée (gauche).

des différentes techniques de stéréovision a été réalisée (voir [45]).

La stéréovision spatiotemporelle (voir [53]) utilise dans sa fonction de coût un voisinage non seulement spatial (les pixels voisins), mais aussi temporel (les mêmes pixels dans plusieurs images de la séquence). Cette technique doit aussi utiliser une méthode d'optimisation globale pour bien fonctionner dans les régions faiblement texturées et n'est donc pas appropriée pour notre système. Enfin, on note que les méthodes basées sur des correspondances de couleurs entre la caméra et le projecteur ne peuvent être utilisées, car les conditions de projection et l'acquisition des images par la caméra altèrent les couleurs de façon non négligeable. En effet, le temps d'exposition de la caméra doit être réduit à $1/30^{e}$ seconde pour qu'elle soit synchronisée avec le projecteur. Pour cette raison, les images de la caméra sont très sombres et leurs couleurs ne correspondent pas à celles de l'image projetée. La figure 1.4 donne un exemple de cette dégradation de contraste.

1.3.2 Lumière structurée

De nombreux systèmes de reconstruction 3D se basent sur la lumière structurée (voir [44]). Cette technique consiste à projeter sur la scène un ou plusieurs patrons de lumière prédéfinis, de façon à identifier par un code unique chaque point de la scène et ainsi chaque pixel de la caméra et du projecteur. La mise en correspondance se fait alors simplement en associant les pixels qui ont le même code. Une grande variété de types de patrons existe. Les méthodes temporelles utilisent des patrons binaires, notamment le code de Gray [28], qui est le type le plus répandu. Il est alors nécessaire de projeter $\lceil \log_2 (\text{largeur}) \rceil$ et $\lceil \log_2 (\text{hauteur}) \rceil$ patrons pour identifier uniquement chaque pixel. D'autres patrons sont conçus de façon à ce qu'à chaque pixel corresponde un voisinage spatial unique. Une étude comparative des types de patrons de lumière structurée a été réalisée (voir [44]).

Dans un système où une projection vidéo (arbitraire) est faite, il n'est pas possible d'interrompre la projection pour recalibrer le système en projetant des patrons de lumière structurée. Une alternative pourrait être de remplacer certaines images de la vidéo (une image par seconde, par exemple) par des patrons de calibration, qui seraient imperceptibles à l'oeil humain. Cette solution n'est toutefois pas applicable. En effet, l'utilisation de patrons temporels exigerait au moins dix images et donc dix secondes pour refaire la mise en correspondance (selon la résolution de la caméra). Les patrons spatiaux ne fonctionneraient pas non plus, car une caméra prenant des images à haute fréquence (généralement 30 par seconde, typiquement) ne donne pas une image suffisamment nette pour les détecter précisément. De même, il est proposé dans [9] d'utiliser les retournements des micromiroirs du projecteur pour encoder les pixels. Cette méthode a toutefois comme inconvénient de modifier légèrement l'intensité des pixels et de réduire l'étendue des couleurs du projecteur, ce qui est un inconvénient majeur dans un contexte artistique. La lumière structurée n'est donc pas une solution adéquate pour des systèmes comme le

nôtre.

1.3.3 Points saillants

Il existe différentes méthodes pour trouver des points saillants dans une image, comme des coins, des lignes, des points. Notamment, la méthode SIFT (*Scale-Invariant Feature Transform* ; voir [32]) permet à la fois de trouver des points saillants et de les faire correspondre dans deux images. SIFT construit une pyramide d'images en prenant la différence entre la convolution de l'image initiale avec deux filtres gaussiens de variance différente. Les points saillants sont identifiés en trouvant les points où se trouve un extremum local aux différents niveaux de la pyramide. Ensuite, à chaque point saillant est associé un vecteur de caractéristiques relatives au voisinage local du point. Ces caractéristiques sont invariantes aux changements d'échelle, d'illumination et à certaines transformations géométriques. Cette approche est fortement inspirée du fonctionnement du cortex visuel chez les humains et les primates (voir [32] pour plus de détails sur la méthode).
SIFT est fréquemment utilisée pour la reconnaissance et le suivi d'objets dans des images ou des séquences vidéo. Toutefois, nos expériences (voir chapitre 4) ont montré que SIFT ne résistait pas bien à de fortes dégradations de contraste ni à de fortes déformations dues à la perspective. Comme le but de notre système est de corriger des déformations dues à la géométrie de la scène, il est impensable d'utiliser une solution qui ne supporte qu'un certain type de scènes. La section 4.3.1 présente les expériences que nous avons faites pour comparer notre méthode avec SIFT.

1.3.4 Marqueurs

Une autre méthode, analogue à la lumière structurée, est de placer dans la scène des marqueurs prédéfinis et facilement détectables. ARTag [16] est un exemple de collection

de tels marqueurs. Les marqueurs étant carrés, l'orientation du plan sur lequel chacun est collé peut être obtenue en analysant la déformation du carré dans la scène. Toutefois, comme pour la lumière structurée, cette méthode est intrusive et ne convient pas à un système comme le nôtre. Ces systèmes de marqueurs sont plutôt utilisés dans le cadre de la réalité augmentée.

Il serait aussi possible d'utiliser des émetteurs et capteurs infrarouge pour identifier des points de la scène. Toutefois, cette technique est non seulement coûteuse et plus complexe d'implantation, mais fonctionne également moins bien dans des environnements où la température peut varier pendant la projection.

CHAPITRE 2

Reconstruction 3D et correction de la distorsion

Ce chapitre décrit en détail LNS, la méthode qui a été développée pour corriger la distorsion induite par la surface pluriplane sur laquelle une séquence vidéo est projetée. Au coeur de cette méthode se trouve une méthode innovatrice de mise en correspondance de pixels, basée sur la cooccurrence statistique d'événements binaires.

2.1 Résumé de la méthode

Le fonctionnement de la méthode LNS se divise en cinq grandes étapes :

1. Calibrage de la caméra et du projecteur. Les paramètres intrinsèques et extrinsèques de chaque appareil sont alors estimés.
2. Projection et acquisition synchronisée d'une séquence vidéo. Une détection de mouvement et une quantification sont faites sur chaque image de chaque séquence. Un profil temporel d'activité-quantification est alors établi pour chaque pixel de la

caméra et du projecteur.

3. Mise en correspondance des pixels de la caméra et du projecteur, basée sur les profils d'activité-quantification. La mise en correspondance est simplifiée par la rectification des images, qui assure que les pixels correspondants sont à la même coordonnée y (voir section 2.3.1).

4. Estimation de la géométrie 3D de la surface de projection, basée sur les correspondances obtenues en 3. L'estimation est faite en décomposant la surface en primitives géométriques (plans, quadriques).

5. Estimation d'une transformation H permettant de corriger la distorsion induite par la surface. Les images subséquentes de la séquence sont transformées par H avant d'être projetées.

Notons que l'étape 5 ne peut être réalisée par notre méthode que si la surface de projection est composée uniquement de plans. Il serait toutefois possible de développer des méthodes plus générales permettant de corriger la distorsion induite par des surfaces quelconques.

2.2 Calibrage du système

Notre système n'est pas constitué de deux caméras comme les autres systèmes de stéréovision classiques, mais d'une caméra et d'un projecteur. Néanmoins, cette différence a peu d'impact sur la méthode de calibrage retenue. Le modèle utilisé est celui du sténopé (*pinhole camera*) avec une projection perspective, tant pour la caméra que pour le projecteur. La projection des points P du monde vers des pixels p sur le plan image d'une

caméra se modélise avec une matrice 3×4 dite de projection.

$$\begin{aligned} p &= JP \\ \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \lambda \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ J_{31} & J_{32} & J_{33} & J_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \end{aligned} \tag{2.1}$$

La matrice J est la matrice de projection et se décompose ainsi :

$$\begin{aligned} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \lambda K \left[R|t\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\ &= \lambda \begin{bmatrix} f_x & [s=0] & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \end{aligned} \tag{2.2}$$

où λ est le facteur requis pour que la 3^e coordonnée de p soit 1. La matrice K contient les paramètres intrinsèques de la caméra. Ceux-ci sont parfois données par le fabricant ; autrement, ils peuvent être estimés. Les paramètres sont :

- (f_x, f_y) : $(\frac{f}{s_x}, \frac{f}{s_y})$;
  - f : la distance focale ;
  - (s_x, s_y) : les dimensions d'un pixel (en unités du monde) ;
- (c_x, c_y) : le centre de projection [1] (en pixels) ;
- s : le facteur de cisaillement.

$s \neq 0$ dans les très rares cas où les pixels ne sont pas rectangulaires. C'est pourquoi nous considérons ici toujours que $s = 0$.

La matrice des paramètres extrinsèques $[R|t]$ se décompose en une matrice de rotation R et un vecteur de translation t. Cette matrice décrit la position de la caméra par rapport à l'origine géométrique du monde. La rotation décrite par la matrice R est quelconque

1. Centre de projection : Le pixel dans l'image où se projette le point de l'espace où est placée la caméra.

et peut être décomposée en rotations autour des trois axes :

$$\begin{aligned} R &= R_x * R_y * R_z \\ R_x &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \\ R_y &= \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \\ R_z &= \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Nous choisissons de faire coïncider l'origine du monde avec la position de la caméra, de façon à avoir, pour la caméra

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \lambda_c K_c [I|0] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

et, pour le projecteur

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \lambda_p K_p [R_p|t_p] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Les matrices K_c, K_p, R_p et le vecteur t_p doivent être connus pour que notre système puisse fonctionner. Généralement, les paramètres intrinsèques sont fournis par le fabricant. Les paramètres extrinsèques peuvent être calculés directement si la position relative de la caméra et du projecteur (y compris leur orientation) est connue avec précision.

Estimation des paramètres La méthode de calibrage du système caméra-projecteur est celle de Zhang [54]. Cette méthode estime la matrice de projection de chaque appareil à l'aide de paires pixel-point 3D. Les détails mathématiques sont à l'annexe A. On y voit entre autres qu'au moins six paires pixel-point 3D sont nécessaires pour estimer la

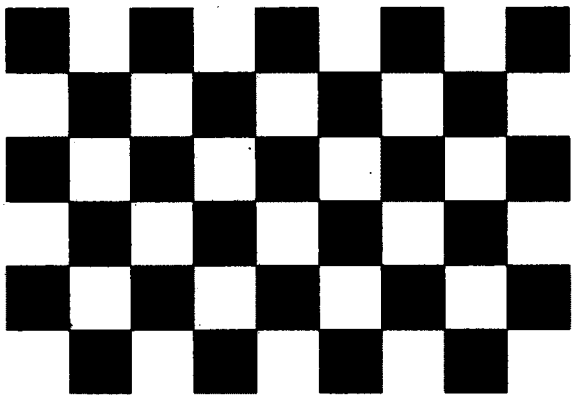


Figure 2.1 – Damier utilisé pour le calibrage de la caméra et du projecteur.

matrice de projection de chaque appareil. Voyons comment obtenir ces paires.
Dans un premier temps, on considère la caméra. Les paires $\{(X, Y, Z), (x_c, y_c)\}$ peuvent être obtenues en plaçant devant la caméra un damier (voir figure 2.1) ou une autre cible de calibrage contenant des points saillants facilement identifiables. Dans le cas du damier, les coins peuvent être détectés dans l'image à l'aide du détecteur de Harris [22]. La position de ces points saillants doit pouvoir être connue avec précision dans le repère du monde. On peut alors associer chaque coin détecté au point 3D qui lui correspond dans la scène.

Lorsque les correspondances $\{(X, Y, Z), (x_c, y_c)\}$ ont été obtenues, on peut chercher celles du projecteur, $\{(X, Y, Z), (x_p, y_p)\}$. Puisque le projecteur ne peut voir la cible de calibration, la même façon de procéder que pour la caméra ne peut être utilisée. On calibre le projecteur en projetant des patrons de lumière structurée sur la cible, comme ceux de la figure 2.2. Ces patrons permettent d'identifier uniquement chaque pixel du projecteur et de la caméra. En associant les pixels de chaque appareil ayant le même code, on obtient des paires $\{(x_p, y_p), (x_c, y_c)\}$. À l'étape de calibrage de la caméra, des paires $\{(X, Y, Z), (x_c, y_c)\}$ avaient été obtenues. En faisant correspondre les paires associées au

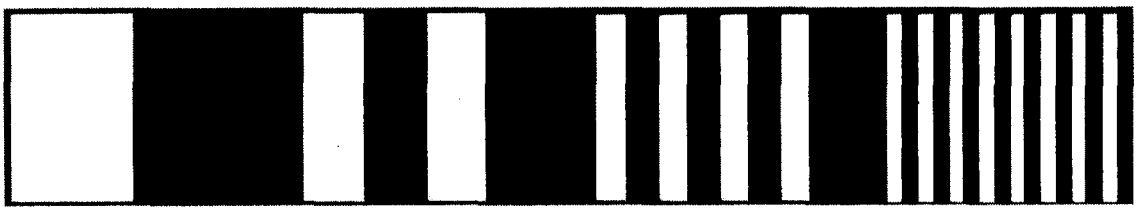


Figure 2.2 – Exemple de patrons binaires de lumière structurée. D'autres patrons similaires (où les bandes verticales seraient de plus en plus fines, jusqu'à avoir une largeur d'un seul pixel) seraient nécessaires pour identifier uniquement chaque pixel.

même pixel (x_c, y_c), on obtient des paires $\{(X, Y, Z), (x_p, y_p)\}$.
On a alors un nombre suffisant de paires pixel-point 3D. Le système d'équations décrit à l'annexe A peut donc être résolu pour la caméra et pour le projecteur.

La phase de calibrage décrite permet d'obtenir les matrices de projection de la caméra et du projecteur. De celles-ci peuvent être obtenues les matrices de paramètres intrinsèques et extrinsèques en faisant une décomposition QR (voir [20]). Alternativement, les paramètres intrinsèques et extrinsèques peuvent être obtenus directement en utilisant la méthode de calibrage de Tsai [51].

2.3 Reconstruction 3D : Méthode par cooccurrences statistiques

Lorsque les paramètres de la caméra et du projecteur sont connus, notre système caméra-projecteur peut se mettre en route. Le projecteur et la caméra sont synchronisés. Ainsi, à chaque image que prend la caméra correspond une et une seule image dans le projecteur. Notre méthode met alors en correspondance les pixels de la caméra et du projecteur afin d'estimer la géométrie 3D de la surface de projection. La mise en correspondance

de pixels se fait à l'aide d'une fonction de coût basée sur la cooccurrence statistique d'événements dans les deux vidéos (voir 2.3.3). Les points 3D obtenus sont enfin divisés en différents objets. Pour chacun de ceux-ci, une transformation visant à corriger la distorsion géométrique induite est appliquée à la partie de l'image qui se projette sur cet objet.

Définitions

Soient I_p la vidéo projetée, $I_p(\cdot,\cdot,t)$ la $t^{\text{ième}}$ image de cette vidéo et $I_p(x_p,y_p,t)$ l'intensité [2] d'un pixel de cette image. De même, soient I_c, $I_c(\cdot,\cdot,t)$ et $I_c(x_c,y_c,t)$ la vidéo, les images de la caméra et leurs pixels. Les pixels de la caméra et du projecteur sont notés (x_c,y_c) et (x_p,y_p) ou plus concisément p_c et p_p. Enfin, l'indice a est utilisé pour représenter tant la caméra que le projecteur dans les cas où les deux appareils sont traités de la même façon.

2.3.1 Géométrie épipolaire

Soient C_c et C_p le centre optique de la caméra et du projecteur. Alors, la droite L_p passant par C_p et p_p se projette sur la droite ℓ_c dans I_c. Cette droite se nomme la droite épipolaire associée à p_p et l'étude des relations de projection entre deux points de vue est la géométrie épipolaire. La figure 2.3 schématise ce concept. La géométrie épipolaire permet de rendre la mise en correspondance plus rapide et de réduire l'erreur de correspondance. En effet, en ne considérant qu'un seul point de vue, on sait que le point P se projetant en p_p est situé quelque part sur L_p (voir figure 2.3). L'identification du pixel correspondant de p_p dans I_c permet de localiser exactement P. Puisque L_p se projette sur ℓ_c et que P est sur L_p, alors p_c doit se trouver sur ℓ_c. Ainsi, étant donné un pixel p_p, la géométrie épipolaire permet de réduire l'espace de recherche de son correspondant p_c à une droite ℓ_c dans I_c, plutôt qu'à I_c entière (voir [23]).

2. Intensité d'un pixel : sa couleur, son niveau de gris

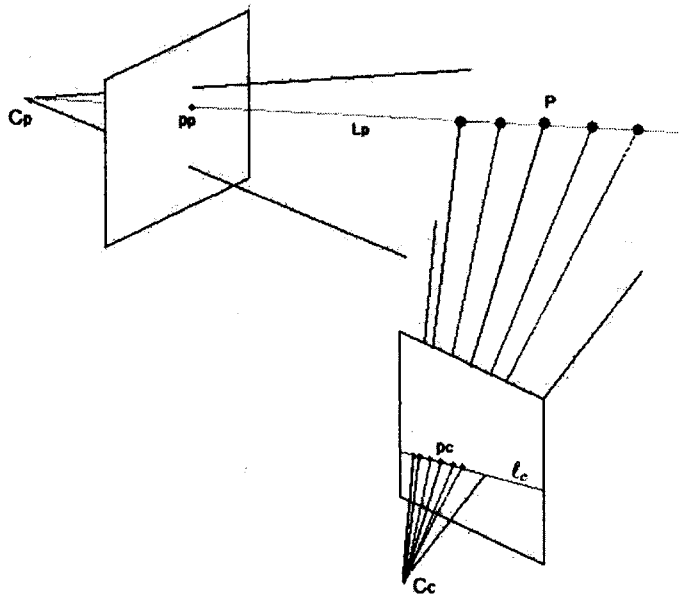


Figure 2.3 – Le rayon de projection L_p se projette sur la droite ℓ_c dans l'image de la caméra. Tous les correspondants potentiels de p_p (en couleur) sont situés sur la droite ℓ_c.

2.3.2 Rectification des images

Une façon courante d'utiliser la géométrie épipolaire est le calcul d'une matrice F, dite fondamentale (voir [34]). Cette matrice met en relation les pixels de la caméra avec ceux du projecteur : $p_p^T F p_c = 0$. Elle peut être estimée, à partir d'un minimum de huit correspondances $p_c \leftrightarrow p_p$ (voir [34], [23]). Ces huit correspondances peuvent être obtenues avec un détecteur de points saillants, comme SIFT [32] ou Harris [22]. La matrice fondamentale peut également être calculée à partir des matrices de paramètres intrinsèques et extrinsèques de la caméra et du projecteur (voir [23]). Toutefois, si le système est calibré et les matrices de paramètres sont connues, il y a une autre façon plus simple d'utiliser la géométrie épipolaire. Cette autre façon, mieux adaptée à notre type d'application, est de s'assurer que les droites épipolaires soient parallèles. Alors, comme l'illustre la figure 2.5, la mise en correspondance d'un pixel (x_c, y_c) de la caméra à un pixel (x_p, y_p) du projecteur se fait à la même coordonnée y. On considère ainsi des correspondances $(x_c, y) \leftrightarrow (x_p, y)$ et il n'est pas nécessaire d'estimer la matrice fondamentale.

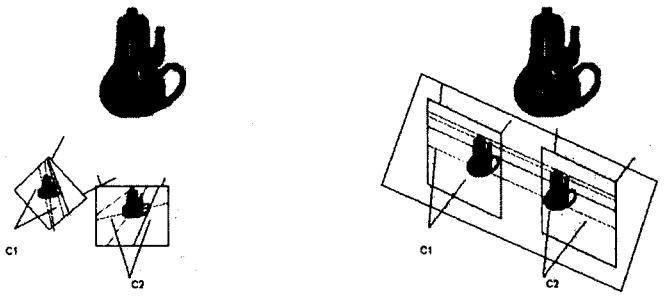


Figure 2.4 – La rectification transforme deux appareils convergents (gauche) en appareils parallèles (droite). Les droites épipolaires, concurrentes dans les images originales, sont parallèles dans les images rectifiées.

Les droites épipolaires sont parallèles dans deux cas : si la caméra et le projecteur sont eux-mêmes parallèles ou si les images sont rectifiées. Comme il est généralement souhaitable de positionner les appareils dans une configuration convergente, afin de s'assurer que la caméra voie toute la projection, les images doivent être rectifiées.
La rectification est un processus par lequel les images acquises par des appareils convergeants sont transformées pour qu'elles semblent acquises par des appareils parallèles. Pour un pixel p_{nr} dans l'image originale non rectifiée et son correspondant p_r dans l'image rectifiée, on a

$$p_r = H_r \cdot p_{nr}$$

où H_r est la matrice de rectification. Les détails du calcul de H_r sont dans l'annexe B. Il est important de noter que généralement, les dimensions des images rectifiées sont différentes de celles des images originales. Par contre, les images rectifiées de la caméra et du projecteur doivent avoir la même résolution en y, et ce même si ce n'était pas le cas à l'origine. À la suite de la rectification, un point donné de la scène doit être à la même coordonnée y dans les deux images.

2.3.3 Cooccurrences statistiques d'événements binaires

Lorsque les images sont rectifiées, un point 3D P de la scène se projette à la même coordonnée y dans les deux images. Ainsi, un pixel (x_c, y) dans l'image rectifiée de la caméra a comme correspondant un pixel situé sur la même ligne dans l'image rectifiée du projecteur, (x_p, y). Une

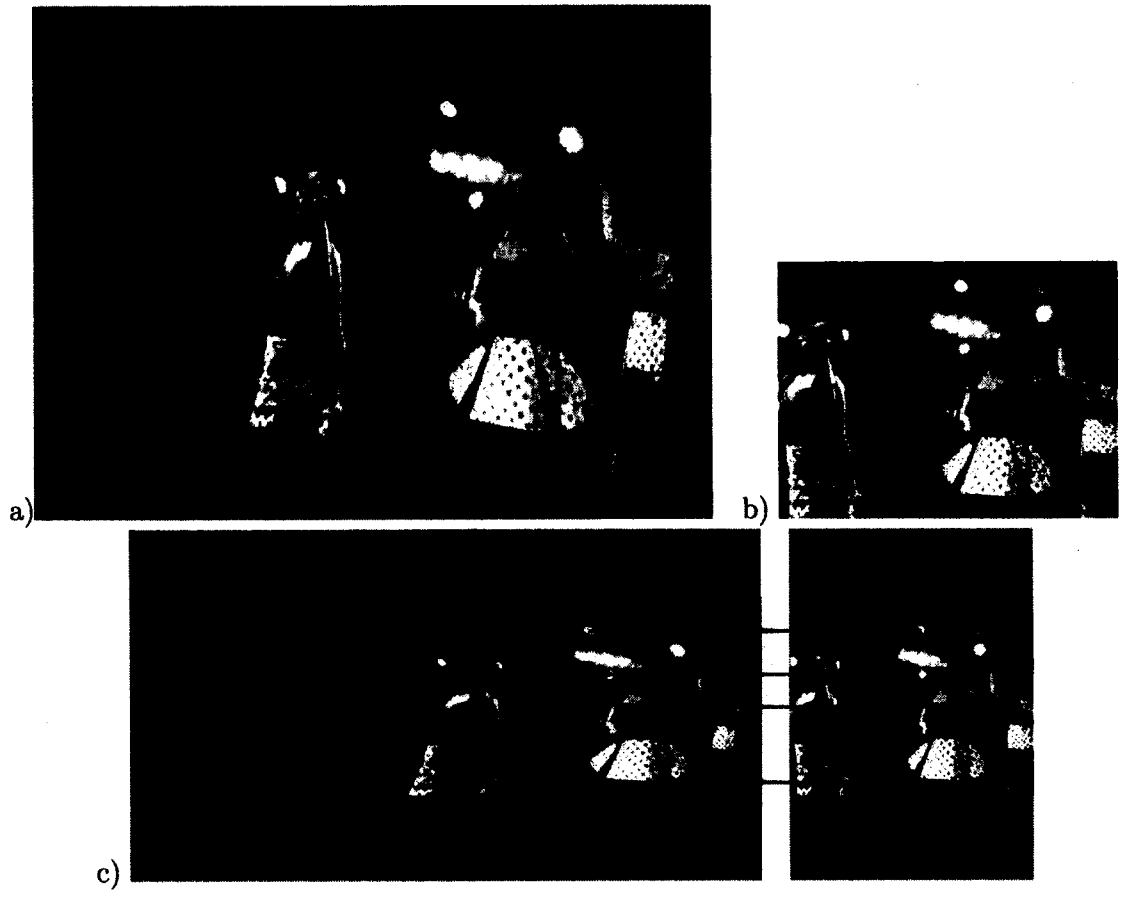


Figure 2.5 – (a) Une image de la caméra et (b) l'image correspondante du projecteur. (c) Après rectification, les correspondances $p_c = (x_c, y) \leftrightarrow p_p = (x_p, y)$ sont à la même coordonnée y dans les deux images.

fonction de coût est néanmoins nécessaire pour établir les correspondances $(x_c, y) \leftrightarrow (x_p, y)$. Tel qu'expliqué en 1.3.1, une fonction de coût basée sur la constance des couleurs ne pourrait fonctionner dans notre cas, en raison du faible contraste des images de la caméra. La figure 2.6 montre des cartes de disparité obtenues avec notre méthode et avec deux méthodes de stéréovision spatiotemporelle utilisant directement l'intensité des pixels. La première fonction de coût est une somme des carrés des différences entre les intensités des pixels :

$$C(p_p, p_c) = \sum_{\tau=t-W+1}^{t} \left[\sum_{i_a, j_a \in \Gamma_a} (I_p(i_p, j_p, \tau) - I_c(i_c, j_c, \tau))^2 \right],$$

où Γ_a est un voisinage local autour de p_a et W la fenêtre temporelle utilisée. La seconde est similaire, mais soustrait l'intensité moyenne du voisinage local avant de calculer les différences :

$$C(p_p, p_c) = \sum_{i_a, j_a \in \Gamma_a} \sum_{\tau=t-W+1}^{t} \left[(I_p(i_p, j_p, \tau) - \overline{I_p(p_p, \tau)}) - (I_c(i_c, j_c, \tau) - \overline{I_c(p_c, \tau)}) \right]^2.$$

Ici, $\overline{I_a(p_a, \tau)}$ est la moyenne des intensités des pixels du voisinage Γ_a. Cette fonction de coût est similaire à la fonction C_3 présentée dans le rapport [14].
Sans dégradation de contraste, les trois méthodes donnent des résultats comparables, mais les deux méthodes de stéréovision spatiotemporelle échouent complètement lorsque l'image de la caméra est moins bien contrastée. Notre fonction de coût se base plutôt sur les cooccurrences statistiques d'événements binaires dans les images du projecteur et de la caméra.

On définit un événement binaire comme étant un changement brusque d'intensité à un pixel donné. Deux pixels de la caméra et du projecteur correspondant à un même point du monde observeront les mêmes événements et auront les mêmes profils d'activité. Avec une quantité suffisante d'images W, notre fonction de coût met en correspondance les pixels dont le profil d'événements est similaire. On rappelle que, dans notre cas, les événements ne sont pas générés par le mouvement des objets de la scène, mais par le mouvement des objets dans la vidéo projetée. La figure 2.7 résume la méthode de mise en correspondance utilisée dans LNS.

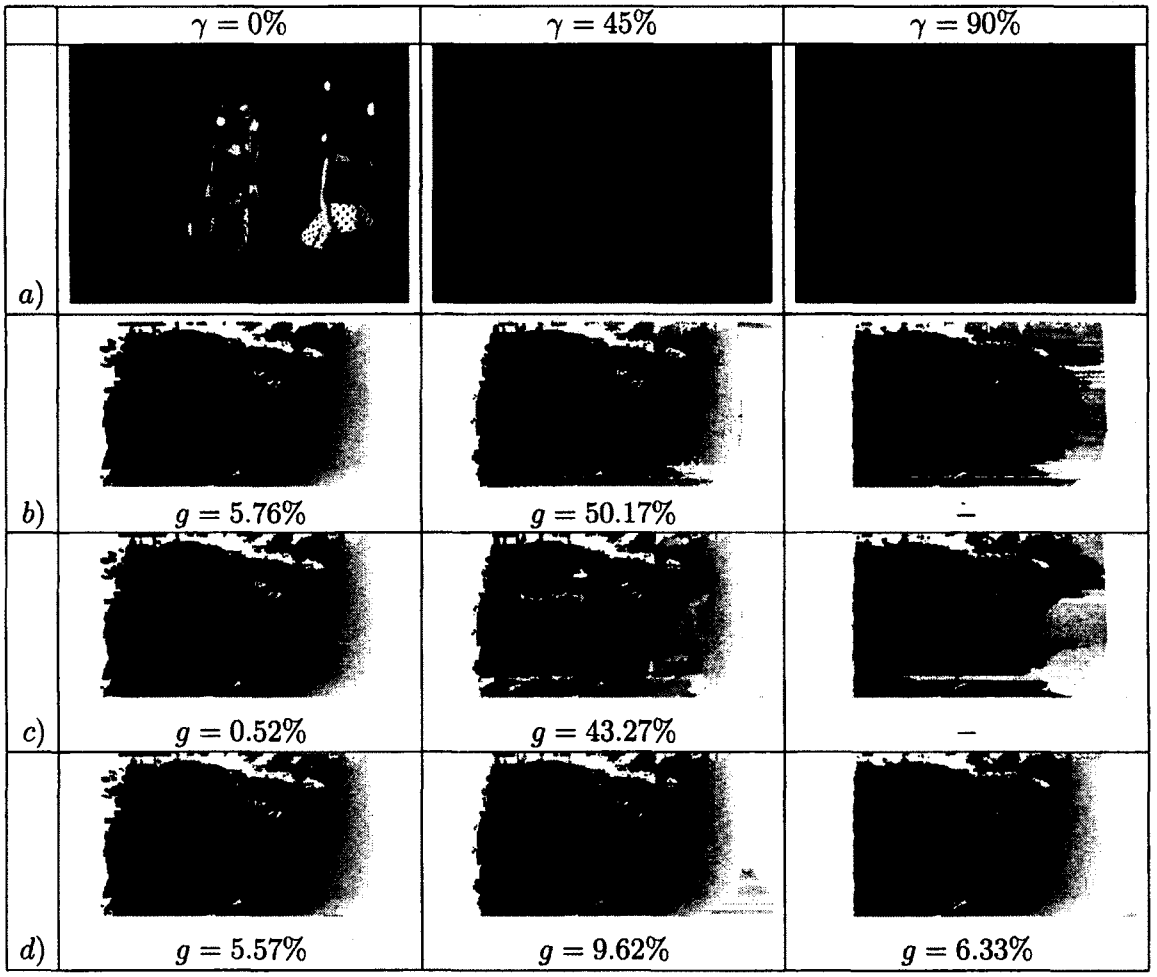


Figure 2.6 – (d) Des cartes de disparité obtenues par notre méthode et par stéréovision spatiotemporelle, avec (c) une somme de différences au carré et (b) une fonction tenant compte de l'intensité locale des pixels. Ces résultats ont été obtenus en projetant $W = 100$ images de la séquence *live band* sur une scène à quatre plans (voir section 4.2). γ est la dégradation de contraste appliquée aux images de la caméra (a) et g le pourcentage de correspondances erronées. Clairement, la stéréovision spatiotemporelle tolère très mal les dégradations de contraste.

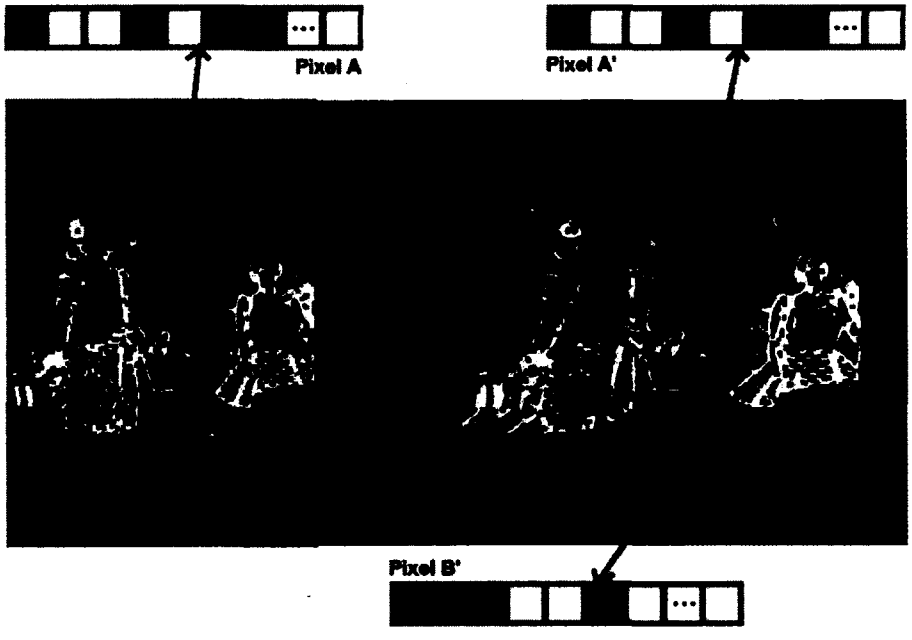


Figure 2.7 – Mise en correspondance par cooccurrences statistiques. Les pixels de la caméra (droite) et du projecteur (gauche) sont mis en correspondance en se basant sur leur profil d'activité respectif. Les pixels A et A' correspondent au même point 3D et ont par conséquent le même profil d'activité. Le pixel B', correspondant à un autre point 3D, a un profil d'activité différent. Les cases noires et blanches représentent la présence et l'absence d'activité dans le temps pour un pixel donné.

Détection de mouvement

Afin de créer les profils d'activité pour chaque pixel du projecteur et de la caméra, nous utilisons une détection de mouvement. La détection de mouvement se fait généralement en prenant la différence entre une image de la séquence et une autre image, dite de référence, contenant uniquement l'arrière-plan. L'image de référence peut simplement être la première image de la séquence ou une image ne contenant aucun objet susceptible d'être en mouvement lors de la séquence. Toutefois, ces méthodes ne tolèrent aucun mouvement de caméra, aucun changement d'illumination et sont restrictives quant à l'activité permise. Par exemple, un objet ajouté à la scène et ne bougeant pas serait toujours considéré comme en mouvement, puisque absent de l'image de référence. Il est donc préférable de mettre l'image de référence à jour à chaque temps t. Ainsi, $I^r(t)$, est une moyenne pondérée de toutes les images précédentes :

$$I_a^r(t) = \begin{cases} I_a(0) & \text{si } t = 0 \\ \alpha * I_a(t-1) + (1-\alpha) * I_a^r(t-1) & \text{si } t > 0 \end{cases}$$

où $\alpha \in [0, 1]$ est un facteur d'apprentissage et l'indice a indique soit le projecteur ou la caméra. On remarque que si $\alpha = 1$, la détection de mouvement est uniquement la différence entre images consécutives, tandis que si $\alpha = 0$, on retrouve le cas où l'image de référence est la première image de la séquence. Étant donnée l'image de référence $I_a^r(t)$, les images binaires de mouvement $V_a(\cdot, \cdot, t)$ sont calculées ainsi :

$$V_a(\cdot, \cdot, t) = \begin{cases} 1 & \text{si } ||I_a(\cdot, \cdot, t) - I_a^r(\cdot, \cdot, t)|| < \tau \\ 0 & \text{sinon} \end{cases} \tag{2.3}$$

où $\tau \geq 0$ est un seuil et $|| \cdot ||$ la norme euclidienne. Il est essentiel d'utiliser un seuil, car les capteurs et les conditions d'acquisition ne sont pas parfaits et peuvent causer de légères variations d'intensité, même dans une scène parfaitement statique. Les pixels tels que $V_a(x_a, y, t) = 1$ sont dits actifs, les autres inactifs.

Seuillage automatique : Méthode d'Otsu S'il apparaît clair qu'il est pertinent d'utiliser un seuil non nul (voir figure 2.8b), la façon de le choisir constitue un sous-domaine de

recherche à part entière (voir [43], [47]). L'efficacité d'un seuil est déterminée entre autres par le contraste observé dans la séquence et les conditions d'illumination lors de la projection (pour les images de la caméra). Comme notre méthode doit pouvoir fonctionner pour n'importe quelle séquence et dans n'importe quelles conditions de projection, nous ne pouvons calculer à l'avance le seuil (voir figure 2.8). Nous devons le calculer *a posteriori*, c'est-à-dire en analysant la séquence.

Pour ce faire, nous utilisons la méthode de segmentation automatique d'Otsu [37]. Cette méthode vise à séparer les pixels d'une image en deux classes les plus cohésives possible. Ainsi, la variance intraclasse doit être la plus petite possible. Otsu démontre dans [37] que la minimisation de la variance intraclasse est équivalente à la maximisation de la variance interclasses (qui est plus simple à calculer). La méthode d'Otsu fait une recherche exhaustive du meilleur seuil τ en sélectionnant celui pour lequel la variance interclasses $\sigma_B(\tau)$ est maximale. La méthode est détaillée dans l'algorithme 1.

Il convient de préciser que la recherche exhaustive du meilleur seuil se fait sur un espace de recherche limité. D'abord, on rappelle que le seuil est comparé à la norme euclidienne de la différence entre deux pixels : $||I_a(t) - I_a^\tau(t)||$. Pour les images à niveaux de gris, $||\cdot|| \Leftrightarrow |\cdot|$. On note que, par souci de simplicité, on ne considère que les seuils entiers. Comme il est certain que $|I_a(x,y) - I_a^\tau(x,y)| \in [0, 255]$, $\tau \in]0, 255[$. Toutefois, l'histogramme de l'image permet généralement de restreindre davantage la plage des valeurs considérées. En effet, si I_a est telle que $I_a(x,y) \in [c_{\min}, c_{\max}]$ (c représentant l'intensité des pixels), alors $\tau \in]0, c_{\max} - c_{\min}[$.
De même, pour les images en couleurs (représentées dans l'espace RGB), $||I_a(x,y) - I_a^\tau(x,y)|| \in [0, \sqrt{255^2 + 255^2 + 255^2}] \approx [0, 442]$ et $\tau \in]0, 442[$.
Enfin, on note que puisque τ dépend de $||I_a(x,y) - I_a^\tau(x,y)||$, τ doit être mis à jour à chaque nouvelle image.
L'intérêt d'utiliser un seuil adaptatif réside dans la grande variabilité et imprévisibilité du niveau de contraste d'une séquence donnée. Or, il est raisonnable de supposer que le contraste puisse varier d'une région à une autre à l'intérieur d'une même image. Ainsi, le seuil calculé risque d'être mieux adapté dans certaines régions d'une image et moins bien ailleurs. Pis encore, un seuil, bien que globalement optimal, pourrait s'avérer localement inefficace partout dans l'image.

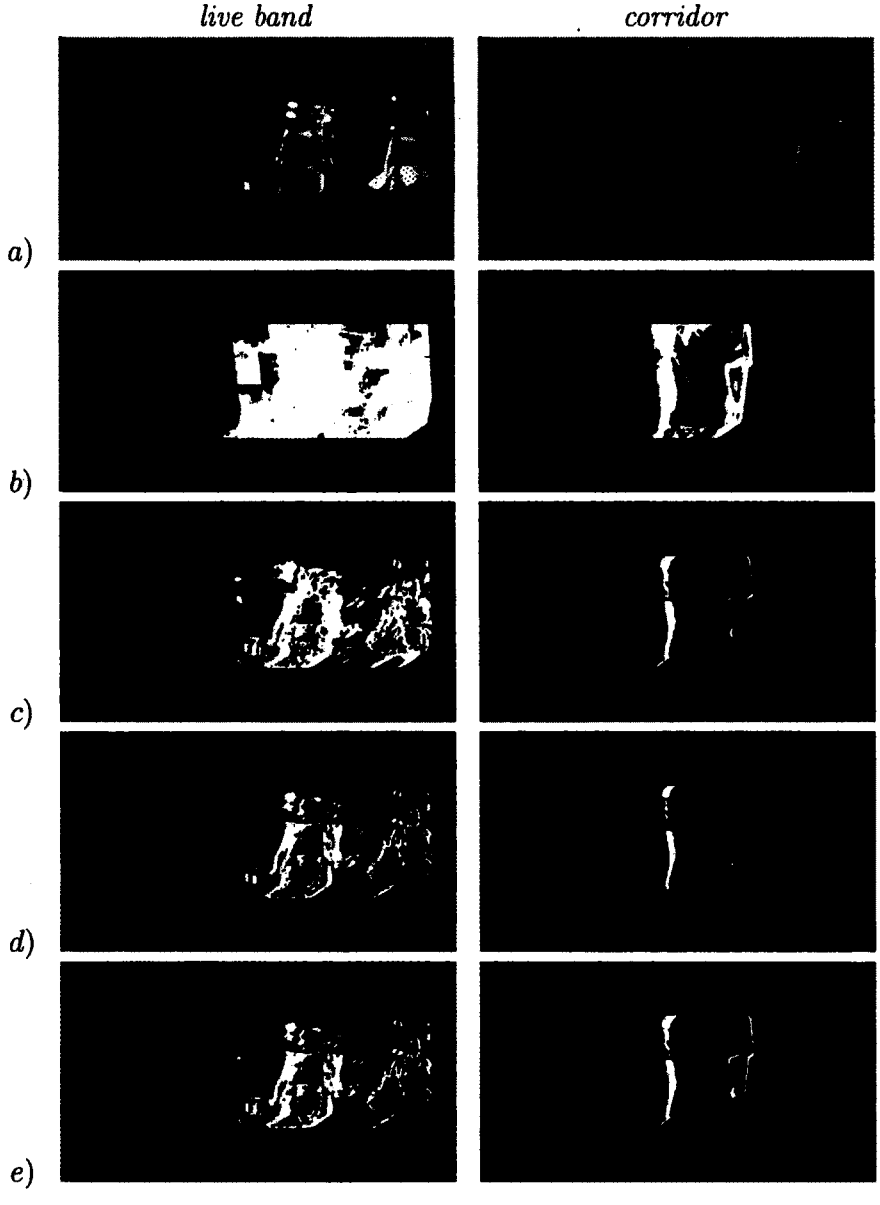


Figure 2.8 – (a) Une image de la séquence vidéo et les images de mouvement correspondant à des seuils de (b) 0, (c) 8, (d) 15 et à (e) la méthode de segmentation de Otsu que nous avons utilisée. Le seuil fixe semblant être le meilleur n'est pas le même pour les deux séquences. La méthode d'Otsu, en revanche, fonctionne bien indépendamment de la séquence. Deux séquences vidéos sont ici présentées.

Algorithme 1 : Calcul du meilleur seuil par la méthode d'Otsu

Entrée : l'image $I = ||I_a(t) - I_a^r(t)||$
Sortie : τ, le seuil optimal

$\hat{\sigma_B^2} \leftarrow 0$
$\mathsf{H} \leftarrow$ `Histogramme` (I)
pour t $\leftarrow 0$ **à** max(H) **faire**
  $\omega_0 \leftarrow \sum_{i=0}^{\tau} \frac{H(i)}{N}$
  $\omega_1 \leftarrow 1 - \omega_0$
  $\mu_0 \leftarrow \sum_{i=0}^{\tau} \frac{i\mathsf{H}(i)}{\omega_0}$
  $\mu_1 \leftarrow \sum_{\tau+1}^{\max(\mathsf{H})} \frac{i\mathsf{H}(i)}{\omega_1}$
  $\sigma_B^2 \leftarrow \omega_0\omega_1(\mu_1 - \mu_0)^2$
  si $\sigma_B^2 > \hat{\sigma_B^2}$ **alors**
    $\hat{\sigma_B^2} \leftarrow \sigma_B^2$
    $\tau \leftarrow$ t
  fin
fin
//Pour les cas où les pixels ne forment qu'une seule classe
si $\tau <$ moyenne (I) **alors** $\tau \leftarrow \max(\mathsf{H})$;

Figure 2.9 – L'image de la caméra, affectée par un bruit gaussien d'écart-type 40 (a). L'algorithme d'Otsu standard trouve énormément de faux positifs (b), alors que notre version les élimine presque complètement (c).

Une façon de remédier à ce problème est de diviser l'image en blocs et de calculer un seuil pour chaque bloc. Ce seuil est optimal pour le pixel situé exactement au centre du bloc ; pour les autres, le seuil est calculé en faisant une interpolation bilinéaire.

Cas particulier L'algorithme d'Otsu fonctionne bien pour segmenter une image en deux classes. Toutefois, il fonctionne moins bien si l'image d'entrée ne contient en fait qu'une classe, c'est-à-dire si les pixels sont tous inactifs ou tous actifs. Le premier cas survient quand la région de la scène où est faite la projection ne couvre pas toute l'image de la caméra. Le second cas survient notamment lorsque l'image contient du bruit (gaussien ou autre) ; alors, les régions où aucune projection n'est faite voient tous leurs pixels varier d'une intensité dépendant du type et de l'intensité du bruit. Otsu tente de séparer ces pixels en deux classes, même si en fait il n'y a aucun mouvement. La figure 2.9 présente un tel cas. Pour parer à cette éventualité, nous nous assurons que le seuil calculé par l'algorithme d'Otsu est plus grand ou égal à la moyenne des pixels du bloc. Dans le cas contraire, nous considérons qu'il n'y a que des pixels inactifs et le seuil utilisé est le seuil maximal possible. Pour davantage de robustesse, nous appliquons un filtre gaussien sur les images au début de la phase de détection de mouvement.

Quantification Notre méthode est avant tout basée sur les cooccurrences statistiques, mais considère également l'intensité des pixels. En effet, les images Q_c et Q_p sont créées à partir des

images de mouvement V_c et V_p de la façon suivante :

$$Q_a(x_a, y, t) = \begin{cases} 0 & \text{si } V_a(x_a, y, t) = 0 \\ \ell(quant(I_a(x_a, y, t), L)) + 1 & \text{si } V_a(x_a, y, t) = 1 \end{cases}$$

où *quant* effectue une quantification à L niveaux sur une version à niveaux de gris de l'image. Cette quantification est faite avec un algorithme de coupe médiane (*median-cut* ; voir [24]). La fonction $\ell \in \{0, 1, ..., L-1\}$ prend l'indice du niveau de gris d'un pixel suite à la quantification. L'indice a réprésente l'un ou l'autre des deux appareils : le projecteur (p) ou la caméra (c).

Fonction de coût

Lorsque les profils d'activité-quantification des pixels de la caméra et du projecteur sont créés, la dernière étape de la mise en correspondance est d'établir quels pixels de la caméra et du projecteur ont les profils les plus similaires, étant donnée une fenêtre temporelle de W images. La façon de procéder est, pour un pixel de la caméra p_c donné, d'associer un coût de correspondance à chaque pixel p_p du projecteur pouvant être son correspondant. Le pixel p_p minimisant ce coût est alors choisi comme correspondant. L'ensemble des p_p considérés est limité d'abord par la contrainte épipolaire ($y \equiv y_c = y_p$). Toutefois, cet ensemble contient des p_p qui engendreraient des points 3D trop près, trop loin ou même derrière la caméra ou le projecteur. Ainsi, en définissant un tronc (*frustum*) $[Z^{min}, Z^{max}]$ où la scène à reconstruire peut être située exclusivement, il est également possible de limiter en x l'espace de recherche (l'annexe C explique comment calculer Z avec x_p et x_c) :

$$\begin{bmatrix} x_p^{min} \\ x_p^{max} \end{bmatrix} = \begin{bmatrix} \frac{t_x}{Z^{max}} + \frac{x_c - c_{x_c}}{f_{x_c}} + c_{x_p} \\ \frac{t_x}{Z^{min}} + \frac{x_c - c_{x_c}}{f_{x_c}} + c_{x_p} \end{bmatrix}$$

La figure 2.10 illustre cette façon de limiter l'espace de recherche de p_p.

Nous pourrions utiliser comme fonction de coût une simple mesure comme la distance euclidienne. Toutefois, cette mesure donne de piètres résultats, car il est fréquent que plusieurs pixels voisins (appartenant à un même objet dans la scène et dans la projection) aient des profils d'activité similaires. Pour éviter de telles ambiguïtés, nous utilisons une fonction de coût

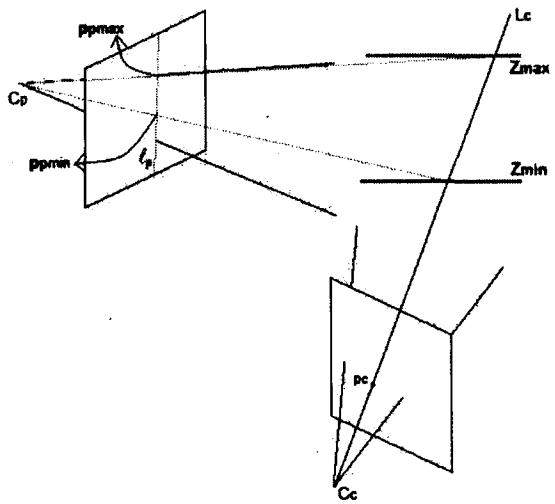


Figure 2.10 – L'objet à reconstruire ne peut être situé que dans la région entre Z_{min} et Z_{max}. Le correspondant de p_c dans l'image du projecteur est alors forcément situé sur un segment de la droite épipolaire ℓ_p, délimité par ${p_p}^{min}$ et ${p_p}^{max}$.

plus complexe qui considère le voisinage local d'un pixel (tant dans l'espace que dans le temps). Cette fonction s'exprime synthétiquement ainsi :

$$C(x_c, x_p, y) = \sum_{\tau=t-W}^{t} \delta(Q_c(x_c, y, \tau), Q_p(x_p, y, \tau)) \tag{2.4}$$

$$\delta(Q_c(x_c, y, \tau), Q_p(x_p, y, \tau)) = \begin{cases} COUT\_MAX & \text{si } p_c \text{ est actif et } p_p \text{ inactif ou vice-versa} \\ \sum_{r \in x,\tau} \omega(\nabla_c^r, \nabla_p^r) & \text{sinon} \end{cases}$$

$$\omega(q_1, q_2) = \begin{cases} COUT\_MAX & \text{si } q_1 * q_2 < 0 \\ 0 & \text{sinon} \end{cases}$$

Ici, r représente l'orientation du gradient spatiotemporel de premier ordre ∇_a. Ainsi, notre fonction de coût pénalise les correspondances pour lesquelles le profil d'activité relatif entre le pixel candidat et ses voisins diffère de la caméra au projecteur. En ce sens, notre fonction de coût fonctionne bien même aux frontières d'objets, où une discontinuité de disparité est observée.

Programmation dynamique La façon la plus simple de minimiser la fonction de coût $C(x_c, x_p, y)$ pour trouver le meilleur correspondant de p_c est de choisir le p_p pour lequel le coût est minimal. Cette méthode vorace se nomme « au plus fort la poche » (*winner-take-all* ou *WTA*; voir [45]). Quoique fort simple et rapide, elle donne généralement de mauvais résultats, établissant trop de correspondances erronées (voir figure 2.11). Pour remédier à ce problème, nous avons ajouté une contrainte d'ordre, implémentée au moyen d'un algorithme de programmation dynamique [3]. Cette contrainte stipule que si $p_c \leftrightarrow p_p$, alors pour un autre pixel $p_c{}'$:

$$\begin{cases} x_c{}' > x_c \\ p_c{}' \leftrightarrow p_p \end{cases} \Rightarrow x_p{}' \geq x_p.$$

La programmation dynamique fonctionne ligne par ligne (sachant que la contrainte épipolaire est respectée). Pour chaque pixel de la caméra, le coût de correspondance avec chaque pixel d'indice $i > 0$ du projecteur est calculé, en tenant compte du coût de correspondance cumulé pour les pixels d'indice 0 à $i - 1$. Ces coûts sont conservés dans un tableau 2D *ct* et les correspondances sont établies en trouvant le chemin de coût minimal allant de la gauche vers la droite dans *ct*. Le remplissage du tableau des coûts est décrit dans l'algorithme 2 et la recherche du chemin de coût minimal dans l'algorithme 3. Le fonctionnement de la programmation dynamique est illustré à la figure 2.12.

Autres méthodes d'optimisation La programmation dynamique n'est pas la seule méthode d'optimisation globale (voir [48]). Notamment, l'optimisation par coupe de graphe (*graph-cut*) ou par propagation de croyances (*belief propagation*) sont aussi fréquemment utilisées pour résoudre des problèmes de mise en correspondance. Ces algorithmes sont fondés sur un graphe dont les noeuds sont les correspondances potentielles entre les pixels de la caméra et du projecteur. Plus de détails sont donnés dans les articles [7], [6] et [30] pour la coupe de graphe, et dans [49] pour la propagation de croyances. Ces méthodes sont très lentes et ne peuvent donc pas être utilisées dans un système comme le nôtre. De plus, elles ne permettent pas d'utiliser une contrainte d'ordre, qui contribue grandement au succès de la programmation dynamique.

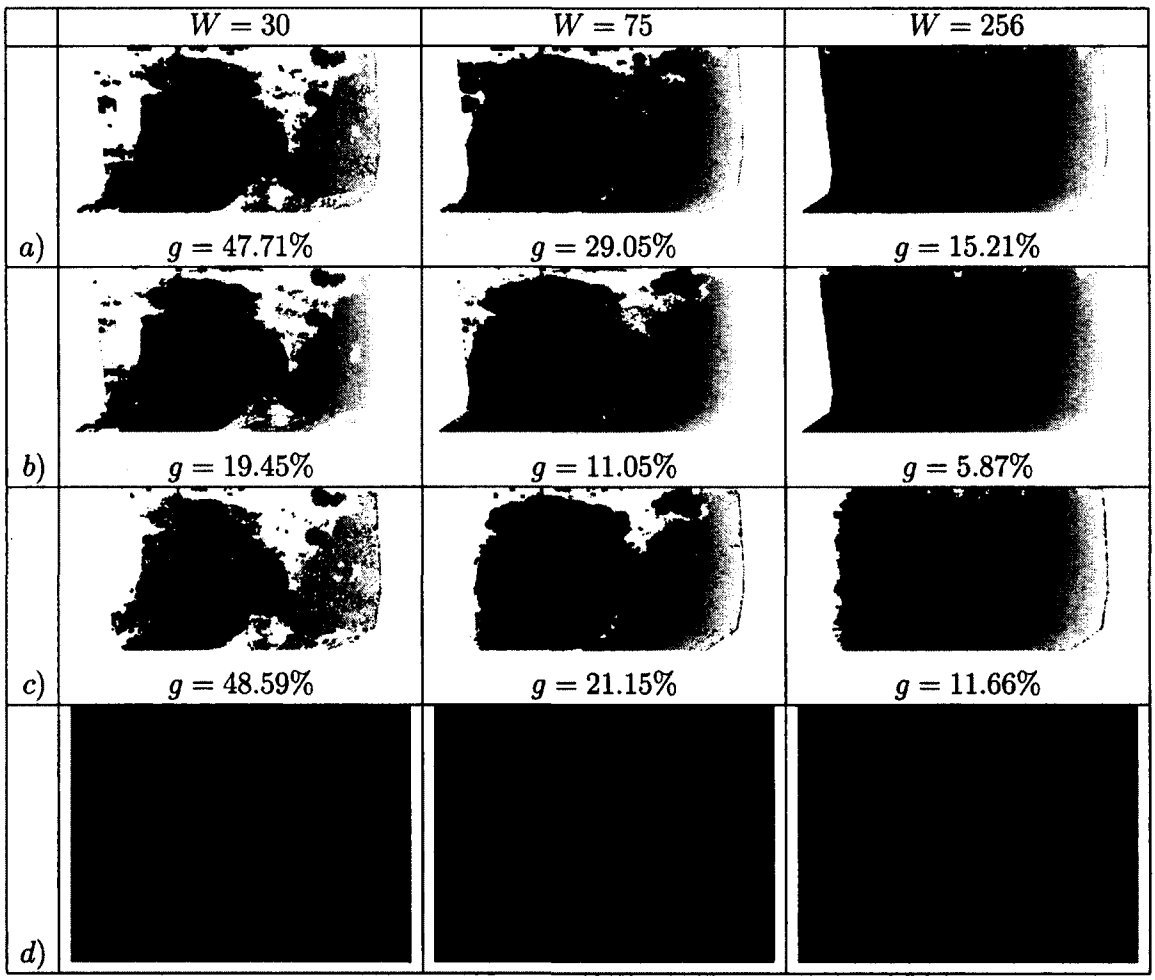


Figure 2.11 – Cartes de disparité obtenues avec WTA (a), la programmation dynamique (b) et la coupe de graphe (c). En (b) et (c), un terme de lissage linéaire a été ajouté à la fonction de coût. g est la proportion de correspondances erronées obtenues. La rangée (d) présente les nuages de points (transformés vers l'espace euclidien) associés aux cartes de disparité en (b).

Algorithme 2 : Programmation dynamique : Calcul du tableau de coûts ct

Entrée : I_c, I_p, une paire d'images rectifiées
Sortie : ct, la table des coûts et ct', la table du chemin de coût minimal

```
1  pour tout y ∈ I_c faire
2      pour tout xproj ∈ I_p (y) faire
3          si actif (I_p, xproj, y) alors
4              ct [0][xproj ] ← C (I_c, I_p, 0, xproj, y)
5              ct' [0][xproj ] ← −1
6          pour tout xcam > 0 ∈ I_c faire
7              [xprojMin, xprojMax ] ← intervalleRecherche (I_c, I_p, xcam, xproj, y)
8              minPrec ← ct.trouverXProjMin (xcam −1)
9              si inactif (I_c, I_p, xcam, xproj, y) alors
10                 pour tout xproj ∈ I_p (y) faire
11                     ct [xcam ][xproj ] ← coutMax
12                     ct' [xcam ][xproj ] ← minPrec
13                 fin
14                 sauter au xcam suivant
15             pour tout xproj ∉ [xprojMin, xprojMax ] faire
16                 ct [xcam ][xproj ] ← coutMax
17                 ct' [xcam ][xproj ] ← minPrec
18             fin
19             pour tout xproj ∈ [xprojMin, xprojMax ] faire
20                 coutPrecMin ← ∞
21                 minX ← −1
22                 pour tout xprojPrec ∈ [xprojMin, xprojMax ] faire
23                     coutPrec ← ct [xcam-1][xprojPrec ]
24                     lissage ← calculerLissage (xcam, xcam −1, xproj, xprojPrec)
25                     si coutPrec + lissage < coutPrecMin alors
26                         coutPrecMin ← coutPrec + lissage
27                         minX ← xprojPrec
28
29                 fin
30                 ct [xcam ][xproj ] ← C (I_c, I_p, xcam, xproj, y) + coutPrecMin
31                 ct' [xcam ][xproj ] ← minX
32             fin
33         fin
34     fin
35 fin
```

Algorithme 3 : Programmation dynamique : Recherche du chemin de coût minimal dans ct

Entrée : ct : la table des coûts ; ct' : la table du chemin de coût minimal ; hauteur, largeur : la hauteur et la largeur de l'image de la caméra

Sortie : S : un ensemble de correspondances (xcam, y, xProjCorr)

```
1  pour tout y ∈ [0,hauteur [ faire
2  |  xcam ← hauteur-1
3  |  xProjCorr ← ct.trouverXProjMin (xcam)
4  |  si actif (xcam, y) alors S.inserer (xcam, y, xProjCorr)
5  |  tant que x>0 faire
6  |  |  xcam ← xcam-1
7  |  |  xProjCorr ← ct' [xcam ][xProjCorr ]
8  |  |  si actif (xcam, y) alors S.inserer (xcam, y, xProjCorr)
9  |  fin
10 fin
```

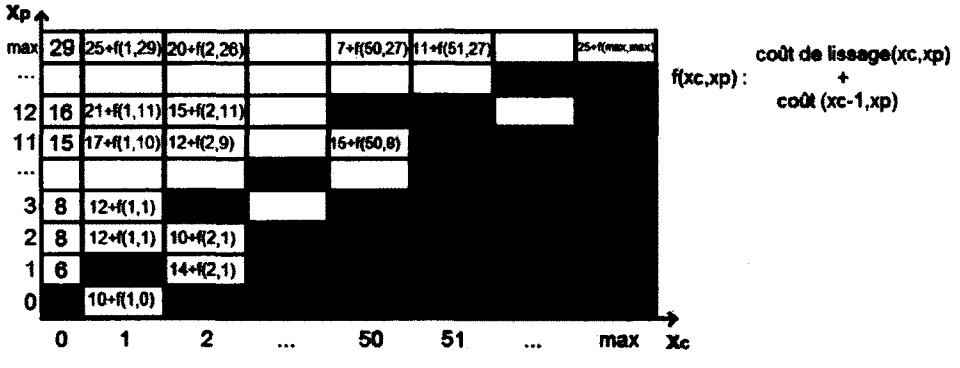


Figure 2.12 – Tableau ct rempli par l'algorithme de programmation dynamique. L'axe horizontal représente les pixels de la caméra et l'axe vertical ceux du projecteur. Chaque case contient la somme du coût de correspondance des pixels x_c et x_p, du plus petit coût cumulatif pour $x_c - 1$, additionné du coût de lissage. Les cases vertes constituent le chemin de coût minimal à travers le tableau. Les cases rouges violent la contrainte d'ordre et ne sont pas considérées.

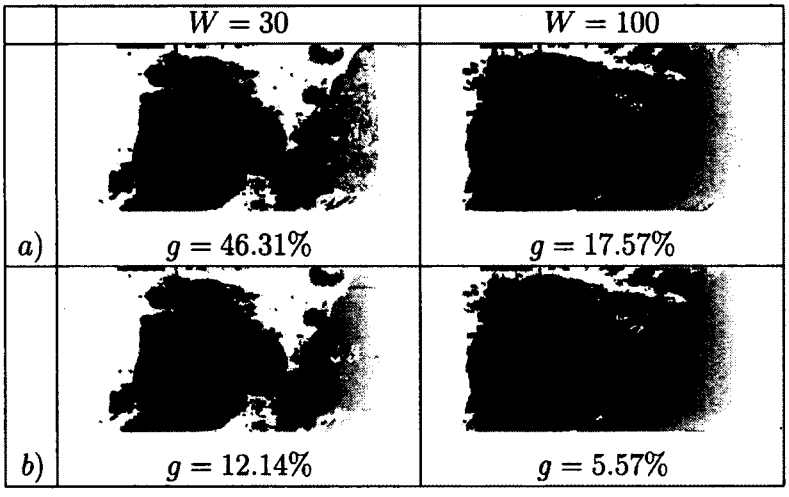


Figure 2.13 – Cartes de disparité obtenues avec lissage en (b) et sans lissage en (a). g est la proportion de correspondances erronées obtenues.

On note toutefois qu'une approche de coupe de graphe adaptée aux occultations [30] utilise une contrainte similaire à la contrainte d'ordre. Néanmoins, tel que mentionné à la section 4.3.7, les résultats obtenus avec notre méthode s'approchent des meilleurs résultats qu'il est théoriquement possible d'obtenir dans notre contexte. L'utilisation d'une méthode d'optimisation plus efficace n'engendrerait pas une amélioration significative des résultats.

Lissage Les méthodes globales d'optimisation rendent possible l'ajout d'un terme de lissage. Ce terme pénalise les changements brusques de disparité entre pixels voisins. En effet, la profondeur des points 3D à la surface d'un objet donné varie généralement de façon progressive et continue. Le terme de lissage permet donc d'éviter des variations subites de disparité, qui causeraient un bruit poivre et sel dans la carte de disparité. Nous utilisons un lissage linéaire, c'est-à-dire que pour deux correspondances potentielles voisines $p_c \leftrightarrow p_{p1}$ et $p_c + 1 \leftrightarrow p_{p2}$, donnant des points 3D de profondeur Z_1 et Z_2, la pénalité ajoutée au coût de correspondance est $|Z_1 - Z_2|$. La figure 2.13 présente des cartes de disparité obtenues avec et sans lissage. On observe de nombreuses discontinuités de disparité sur celle obtenues sans lissage, alors que celles obtenues avec le terme de lissage en sont exemptes.

2.4 Estimation de primitives

Lorsque la mise en correspondancè est terminée, nous avons à notre disposition un ensemble S de points (x_c, y, x_p), dans un espace dit projectif (voir section 3.5). La figure 2.11 montre que plus la fenêtre temporelle W est grande, plus S est dense. Toutefois, pour que le système puisse s'adapter rapidement, W doit rester petit (2 secondes ou moins) et S est alors clairsemé. Pour cette raison, il n'est pas possible d'établir une correspondance pixel-point 3D pour tous les pixels du projecteur. Il est donc nécessaire de faire une approximation des objets de la scène par des primitives géométriques[3] afin de pouvoir corriger la distorsion. Selon Hakala et al. [21], 85 % des objets rencontrés dans des scènes industrielles peuvent être décomposés en plans, sphères, cylindres ou en cônes. Nous avons principalement travaillé avec les plans et les quadriques. Nous avons étudié différentes approches pour estimer les primitives. Nous avons retenu RANSAC, qui s'est clairement démarqué comme étant la méthode la plus efficace. Nos travaux sur l'estimation de primitives sont présentés au chapite 3.

2.5 Correction de la distorsion géométrique

La distorsion induite par la surface de projection est corrigée en transformant les images de la séquence vidéo avant de les projeter. Notons que la transformation à appliquer se décompose en sous-transformations, chacune étant associée à une des primitives de la scène et appliquée sur la région de l'image correspondante. Lorsque les primitives de la scène sont connues, il est possible de calculer les transformations locales à appliquer aux images du projecteur.

Transformation de l'image du projecteur Soit I_p une image du projecteur (originale) et $I_p{}'$ l'image I_p, transformée de façon à éliminer la distorsion géométrique induite par la scène. Soient également I_v et $I_v{}'$ la projection des images I_p et $I_p{}'$ telles que déformées par la scène et

3. Primitives géométriques : Formes géométriques ne pouvant être décomposées. Les plans, sphères, cylindres, cônes et tores sont les principales primitives géométriques.

observées par un point de vue v donné[4].

L'objectif est alors que

$$I_v{}' = I_p, \tag{2.5}$$

mais initialement, on a plutôt

$$I_v = HI_p \tag{2.6}$$

$$I_v{}' = hI_p{}', \tag{2.7}$$

où h est la déformation induite par la scène. h définit une association entre les pixels de l'image I_p et des points 3D de la scène, et est donc inversible[5] En utilisant les équations 2.5 et 2.7, on obtient une expression pour $I_p{}'$:

$$\begin{aligned} hI_p{}' &= I_p \\ I_p{}' &= h^{-1}I_p \end{aligned}$$

Calcul de H (cas de plans) Si la surface de projection est plane ou pluriplane, H est une matrice 3×3, dite d'homographie, telle que

$$p_v = Hp_p, \tag{2.8}$$

où p_v et p_p sont respectivement des points de l'image du point de vue donné et du projecteur, en coordonnées homogènes. H se calcule en utilisant la géométrie projective.

Supposons sans perte de généralité que la scène ne soit compososée que d'un seul plan, $\pi = (\vec{n}, d)$, où $\vec{n}$ le vecteur normal du plan et d sa distance de l'origine. Soit $P = (X, Y, Z, 1)$ un point de la scène, situé sur π. Tel que vu à la section 2.2, la projection d'un point de la scène sur un pixel d'une image est définie par l'équation

$$p = \lambda K[R|t]P,$$

4. La correction de la distorsion est possible uniquement pour un point de vue donné.

5. h n'est pas inversible si la scène contient des points à l'infini. Toutefois, puisqu'il est impossible de projeter des images sur de tels points, h est ici considérée toujours inversible.

où K et $[R|t]$ sont les matrices de paramètres intrinsèques et extrinsèques. Si le projecteur est placé à l'origine, on a

$$\begin{aligned} p_p &= \lambda K_p [I|0] P \\ &= \lambda K_p P. \end{aligned}$$

P s'exprime donc en fonction de p_p de la façon suivante

$$P = \frac{1}{\lambda} {K_p}^{-1} p_p.$$

Rappelons que le point $P = (\frac{1}{\lambda} {K_p}^{-1} p_p, 1)$ est exprimé en coordonnées homogènes. Soit $< a, b >$ le produit scalaire entre les vecteurs a et b. P étant sur π, on a

$$\begin{aligned} < P, \pi > &= 0 \\ \frac{< {K_p}^{-1} p_p, \vec{n} >}{\lambda} + d &= 0 \\ \lambda &= -\frac{< {K_p}^{-1} p_p, \vec{n} >}{d}. \end{aligned}$$

Projetons P sur v, le point de vue de l'observateur. La position de ce point de vue est définie par la matrice $[R|t]$, dans le repère du projecteur. L'opérateur * représente ici un produit tensoriel entre deux vecteurs dont le résultat est une matrice. On a :

$$\begin{aligned} p_v &= \lambda K_v [R|t] P \\ &= \lambda K_v [R|t] (\frac{1}{\lambda} {K_p}^{-1} p_p, 1)^T \\ &= K_v [R|t] ({K_p}^{-1} p_p, \lambda)^T \\ &= K_v [R|t] ({K_p}^{-1} p_p, -\frac{< {K_p}^{-1} p_p, \vec{n} >}{d})^T \\ &= K_v (R \cdot ({K_p}^{-1} p_p) - \frac{< {K_p}^{-1} p_p, t * \vec{n} >}{d})^T \\ p_v &= K_v (R - \frac{1}{d} t * \vec{n}) {K_p}^{-1} p_p. \end{aligned} \tag{2.9}$$

Notons qu'ici, $t * \vec{n}$ est une matrice 3×3. Des équations 2.8 et 2.9, on tire

$$H = K_v (R - \frac{1}{d} t * \vec{n}) {K_p}^{-1}.$$

CHAPITRE 3

Estimation de primitives géométriques

De grands efforts de recherche ont été consacrés à l'estimation de primitives, ou plus généralement à l'estimation de modèles. Notamment, la communauté de la télédétection (*range sensing*) a fourni des efforts considérables en ce sens (voir [5], [8], [19], [40], [50], [52]). La plupart des méthodes proposées appartiennent à l'une de trois grandes familles d'approches [1] : les régressions, la transformée de Hough et les optimiseurs stochastiques (par exemple, RANSAC).

Ce chapitre traite des façons les plus courantes d'estimer des primitives géométriques à partir d'un ensemble d'observations. Les principales méthodes sont décrites en détail, de même que leur implémentation dans le cadre de notre méthode. Enfin, nous présentons également une méthode d'estimation de plans fondée sur l'algorithme des nuées dynamiques (*k-means* ; voir [35]) avec laquelle nous avons expérimenté.

1. On trouve également des méthodes orphelines, dont celle de Hoppe et al. [25], qui estime des surfaces générales sans les décrire mathématiquement. Cette méthode trouve le plan tangent à chaque point pour définir une fonction de distance et utilise des contours actifs pour créer la surface.

3.1 Régressions

La méthode des moindres carrés, aussi appelée « régression », est une façon analytique de trouver exactement les paramètres d'un modèle qui cadrent le mieux avec un ensemble d'observations potentiellement bruitées. Soit le modèle exact défini par une fonction $f(X_i) = 0$. Par exemple, dans le cas d'une droite, $f \equiv ax_i + b - y_i$ et les observations X_i sont des couples (x_i, y_i). Puisque les observations X_i peuvent être bruitées (pas exactement sur la droite, mais près), l'égalité n'est en pratique jamais respectée et on a plutôt $f(X_i) = \epsilon_i$ (voir figure 3.5a). L'objectif de la méthode est de trouver les paramètres de f qui minimisent l'erreur ϵ. Il s'agit donc d'un problème de minimisation, qui se résout en posant le gradient ∇ de l'expression égal à 0 : $\nabla(\sum_{i=1}^{n}(f(X_i) - \epsilon_i)^2) = 0$, puis en résolvant le système d'équations obtenu. On utilise le carré de $f(X_i) - \epsilon_i$ pour deux raisons : pour éviter que des ϵ positifs et négatifs ne s'annulent et parce que la fonction d'exponentiation est dérivable en tout point, contrairement à la valeur absolue.

3.1.1 Estimation de plans

On définit un plan π par son vecteur normal $\vec{n} = (n_x, n_y, n_z)$ et sa distance d de l'origine. Son équation est alors $\pi : n_x x + n_y y + n_z z + d = 0$. À priori, quatre paramètres doivent donc être estimés, mais puisqu'une surface de projection ne peut être parallèle à l'axe optique de la caméra[2], il nous est possible de fixer $n_z = 1$. On a donc $\pi = (n_x, n_y, 1, d)$ et on note $\hat{\pi} = (n_x, n_y, d)$. Pour un point $P = (X, Y, Z, 1)$ situé sur le plan π, on a $\pi P = 0$. Or, puisque la position du point peut être inexacte en raison de bruit, on a plutôt

$$\pi P = \epsilon_P \tag{3.1}$$

2. Un plan parallèle à l'axe optique de la caméra ne présente à celle-ci aucune surface où l'image peut être observée. Par exemple, il n'est pas possible de projeter une image sur la tranche d'une feuille de papier, puisqu'elle est infiniment petite. Ainsi, il est certain que $n_z \neq 0$.

On note au passage que ϵ_P est la distance algébrique entre un point et le plan. Les paramètres du plan s'obtiennent en minimisant, pour un ensemble de points S, la somme des carrés des ϵ_P :

$$\begin{aligned} \epsilon_P &= \sum_{P \in S} (n_x X + n_y Y + Z + d)^2 \\ &\equiv K(n_x, n_y, 1, d) \\ \Rightarrow \pi &= \arg\min_{\pi_i} K \end{aligned}$$

Le point minimum de K est atteint quand son gradient est nul :

$$\begin{aligned} \nabla K = \vec{0} \Rightarrow & \begin{cases} \frac{\partial K}{\partial n_x} \\ \frac{\partial K}{\partial n_y} \\ \frac{\partial K}{\partial d} \end{cases} \\ \Rightarrow & \begin{cases} 2 \sum_{P \in S} X(n_x X + n_y Y + Z + d) \\ 2 \sum_{P \in S} Y(n_x X + n_y Y + Z + d) \\ 2 \sum_{P \in S} (n_x X + n_y Y + Z + d) \end{cases} \\ \Rightarrow & \begin{cases} 2 \sum_{P \in S} (n_x X^2 + n_y XY + XZ + Xd) \\ 2 \sum_{P \in S} (n_x XY + n_y Y^2 + YZ + Yd) \\ 2 \sum_{P \in S} (n_x X + n_y Y + Z + d) \end{cases} \end{aligned} \tag{3.2}$$

Le système 3.2 peut également s'exprimer sous forme matricielle :

$$\begin{aligned} \sum_{P \in S} \begin{bmatrix} X^2 & XY & X \\ YX & Y^2 & Y \\ X & Y & 1 \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ d \end{bmatrix} &= -\sum_{P \in S} \begin{bmatrix} XZ \\ YZ \\ Z \end{bmatrix} \\ M\hat{\pi} &= B \end{aligned}$$

et ainsi $\hat{\pi} = M^{-1}B$.

Cas particulier Si S contient exactement trois points, il existe une méthode plus simple pour calculer π. En définissant deux vecteurs $v_1 = P_1 - P_2$ et $v_2 = P_1 - P_3$, le vecteur normal est le produit vectoriel : $\vec{n} = v_1 \times v_2$. d s'obtient ensuite directement avec l'un des points : $d =< -P_i, \vec{n} >$. Enfin, on note qu'il n'est pas possible de trouver un plan si les points de S sont tous colinéaires.

3.1.2 Estimation de quadriques

Une quadrique est une surface courbe générale dans l'espace. Les sphères, les hyperboloïdes, les paraboloïdes, les cônes, etc. en sont des cas particuliers. Mathématiquement, une quadrique est définie par dix paramètres :

$$\underbrace{\begin{bmatrix} x & y & z & 1 \end{bmatrix}}_{P^T} \underbrace{\begin{bmatrix} a & e & f & h \\ e & b & g & i \\ f & g & c & j \\ h & i & j & d \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{P} = 0.$$

En développant, on obtient l'équation suivante :

$$ax^2 + by^2 + cz^2 + d + 2(exy + fxz + gyz + xh + yi + zj) = 0.$$

Rouhani et Sappa [41] décrivent comment faire des régressions linéaires et non linéaires pour estimer les paramètres de quadriques et proposent une nouvelle façon de mesurer l'erreur d'estimation.

Minimisation de la distance algébrique

Comme dans le cas des plans, on considère un ensemble S de points situés sur la quadrique, mais étant potentiellement bruités. On a donc

$$ax^2 + by^2 + cz^2 + d + 2(exy + fxz + gyz + xh + yi + zj) = \epsilon_P$$

et on cherche la quadrique Q telle que la somme des carrés des ϵ_P est minimale.

$$\begin{aligned} \epsilon &= \sum_{P \in S} \epsilon_P^2 \\ &\equiv K(a,b,c,d,e,f,g,h,i,j) \\ \Rightarrow Q &= \arg\min_{Q_i} K \end{aligned}$$

On minimise ϵ en posant $\nabla K = \vec{0}$. Q est trouvé en résolvant le système obtenu :

$$\sum_{i \in S} \begin{bmatrix} x_i^4 & x_i^2 y_i^2 & x_i^2 z_i^2 & x_i^2 y_i z_i & x_i^3 z_i & x_i^3 y_i & x_i^3 & x_i^2 y_i & x_i^2 z_i & x_i^2 \\ & y_i^4 & y_i^2 z_i^2 & y_i^3 z_i & y_i^2 x_i z_i & x_i y_i^3 & x_i y_i^2 & y_i^3 & y_i^2 z_i & y_i^2 \\ & & z_i^4 & y_i z_i^3 & x_i z_i^3 & x_i y_i z_i^2 & x_i z_i^2 & y_i z_i^2 & z_i^3 & z_i^2 \\ & & & y_i^2 z_i^2 & x_i y_i z_i^2 & x_i y_i^2 z_i & x_i y_i z_i & y_i^2 z_i & y_i z_i^2 & y_i z_i \\ & & & & x_i^2 z_i^2 & x_i^2 y_i z_i & x_i^2 z_i & x_i y_i z_i & x_i z_i^2 & x_i z_i \\ & & & & & x_i^2 y_i^2 & x_i^2 y_i & x_i y_i^2 & x_i y_i z_i & x_i y_i \\ & & & & & & x_i^2 & x_i y_i & x_i z_i & x_i \\ & & & & & & & y_i^2 & y_i z_i & y_i \\ & & & & & & & & z_i^2 & z_i \\ & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ f \\ g \\ h \\ i \\ j \\ k \\ d \end{bmatrix} = \vec{0}$$

$$\beta Q = \vec{0}.$$

Notons que la matrice β est symétrique et que seuls les éléments de sa moitié supérieure sont indiqués ici. Ce système est homogène (un côté de l'égalité est nul) ; on doit donc utiliser une décomposition en valeurs singulières pour le résoudre (voir annexe A.0.8).

Minimisation de la distance géométrique

Cette méthode a par contre l'inconvénient de minimiser la distance algébrique entre les points et la quadrique. Minimiser la distance géométrique est une tâche considérablement plus complexe. En effet, puisque les paramètres de la quadrique ne sont pas connus, il est nécessaire d'utiliser un processus itératif. Dans un premier temps, les paramètres sont estimés, puis la distance des points

de S à la quadrique est calculée. Trouver cette distance requiert également un processus itératif, car la non-linéarité des quadriques rend impossible son calcul direct. Nous utilisons plutôt une méthode approximant la distance géométrique, la mesure d'erreur de Sampson. Cette mesure se base sur un développement en série de Taylor du premier ordre de l'équation de la quadrique autour du point $\hat{P}$, le point sur la quadrique le plus près d'un point P. Hartley et Zisserman [23] montrent que, dans notre cas, l'erreur de Sampson se calcule ainsi :

$$||\delta_P||^2 = \frac{\epsilon_P^2}{JJ^t}$$

où J est le jacobien (matrice des dérivées partielles) de la quadrique. Rappelons que, dans le cadre du calcul de la distance géométrique, les paramètres de la quadrique sont fixés et ce sont les composantes de $\hat{P}$ qui sont les variables. On a donc :

$$\begin{aligned} J &= \left[\frac{\partial(P^TQP)}{\partial x} \quad \frac{\partial(P^TQP)}{\partial y} \quad \frac{\partial(P^TQP)}{\partial z} \right] \\ &= \left[2P^TQ \cdot (1,0,0,0)^t \quad 2P^TQ \cdot (0,1,0,0)^t \quad 2P^TQ \cdot (0,0,1,0)^t \right] \\ &= \left[2(ax+ey+fz+h) \quad 2(ex+by+gz+i) \quad 2(fx+gy+cz+j) \right] \end{aligned}$$

et conséquemment :

$$||\delta_P||^2 = \frac{(P^TQP)^2}{4 * [(ax+ey+fx+h)^2 + (ex+by+gz+i)^2 + (fx+gy+cz+j)^2]} \tag{3.3}$$

La recherche itérative de Q minimisant $||\delta_P||^2$ se fait avec l'algorithme de minimisation non linéaire de Levengberg-Marquardt [39]. Q est initialisée en minimisant sa distance algébrique aux points de S (voir 3.1.2).

3.1.3 Estimation de sphères

Bien qu'en théorie, il soit possible d'estimer l'équation de tout type de quadrique, en pratique, cette méthode s'avère peu efficace pour les cas particuliers où plusieurs des paramètres doivent être égaux à 0 ou à 1 (notamment la sphère). En effet, l'estimation n'étant par définition pas un calcul exact, ces paramètres ne prennent pas exactement les valeurs requises, ce qui fait qu'une

Figure 3.1 – L'estimation de quadriques, même dans un cas simple (une seule primitive, sans bruit), ne réussit pas à bien retrouver la sphère. En a) et b), elle trouve des ellipsoïdes de différentes dimensions, alors qu'en c), elle obtient plutôt un hyperboloïde à deux nappes. Dans tous les cas, la quadrique retrouvée passe pourtant par les points de S, ou sinon très près.

surface devant par exemple être une sphère peut prendre une toute autre allure. De plus, S ne contient des points que sur une petite portion de la surface de la sphère (là où les images sont projetées) et ces points décrivent approximativement plusieurs quadriques différentes. La figure 3.1 illustre ces deux phénomènes. Plusieurs approches ont été développées pour estimer directement certains types de quadriques. Notamment, Lukacs et al. [33] proposent des méthodes de régression non linéaire pour estimer directement des sphères, des cylindres, des cônes et des tores. Il est ici décrit comment utiliser une régression linéaire pour estimer une sphère.

La sphère est définie par son centre $C = (c_x, c_y, c_z)$ et son rayon r. Elle est un cas particulier de

quadrique où

$$\begin{aligned} a &= 1 \\ b &= 1 \\ c &= 1 \\ d &= c_x^2 + c_y^2 + c_z^2 - r^2 \\ e &= 0 \\ f &= 0 \\ g &= 0 \\ h &= c_x \\ i &= c_y \\ j &= c_z \end{aligned}$$

Toutefois, on ne réprésente généralement pas la sphère comme une quadrique, mais plutôt avec l'équation simplifiée :

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z) - r^2 = 0$$

En développant, on obtient

$$x^2 + y^2 + z^2 - 2(c_x x + c_y y + c_z z) + c_x^2 + c_y^2 + c_z^2 - r^2 = 0$$

Les paramètres à estimer sont ici $Q = (c_x, c_y, c_z, r)$. On remarque qu'en utilisant la technique habituelle de minimisation, les dérivées partielles ont une forme plutôt plus complexe. Avec $K = \sum_{P \in S} \epsilon_P^2$, on a

$$\nabla K = \begin{bmatrix} \sum_{P \in S} \epsilon_P \cdot (-2x + 2c_x) \\ \sum_{P \in S} \epsilon_P \cdot (-2y + 2c_y) \\ \sum_{P \in S} \epsilon_P \cdot (-2z + 2c_z) \\ \sum_{P \in S} \epsilon_P \cdot (-2r) \end{bmatrix} = \vec{0}$$

Cette équation ne peut être transformée en un système d'équations linéaires comme c'était le cas pour les plans et les quadriques. Toutefois, il est possible de définir de nouvelles variables A, B, C, D :

$$\begin{cases} A = -2c_x \\ B = -2c_y \\ C = -2c_z \\ D = c_x^2 + c_y^2 + c_z^2 - r^2 \end{cases} \tag{3.4}$$

et $\hat{Q} = (A, B, C, D)$, de sorte que

$$\begin{aligned} K' &= \sum_{P \in S} x^2 + y^2 + z^2 + Ax + By + Cz + D \\ \nabla K' = \vec{0} &= \begin{bmatrix} \sum_{P \in S} (x^2 + y^2 + z^2 + Ax + By + Cz + D) \cdot x \\ \sum_{P \in S} (x^2 + y^2 + z^2 + Ax + By + Cz + D) \cdot y \\ \sum_{P \in S} (x^2 + y^2 + z^2 + Ax + By + Cz + D) \cdot z \\ \sum_{P \in S} (x^2 + y^2 + z^2 + Ax + By + Cz + D) \end{bmatrix} \end{aligned} \tag{3.5}$$

L'équation 3.5 est équivalente au système suivant :

$$\sum_{P \in S} \begin{bmatrix} x^2 & xy & xz & x \\ xy & y^2 & yz & y \\ xz & yz & z^2 & z \\ x & y & z & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = -\sum_{P \in S} \begin{bmatrix} x^3 + xy^2 + xz^2 \\ x^2y + y^3 + yz^2 \\ x^2z + y^2z + z^3 \\ x^2 + y^2 + z^2 \end{bmatrix}$$

$$\begin{aligned} M\hat{Q} &= N \\ \hat{Q} &= M^{-1}N. \end{aligned}$$

Q peut être retrouvée à partir de $\hat{Q}$ à l'aide des relations en 3.4.

3.2 Transformée de Hough

La transformée de Hough ([26]) est une méthode de détection de droites dans des images. Elle se base sur la dualité entre les points et les droites : une droite est composée d'une infinité de points, mais par un point passent une infinité de droites. Une droite est définie par sa pente a et son ordonnée à l'origine b : $y = ax + b$. La transformée de Hough fait passer les points de l'espace euclidien $\{x, y\}$ habituel à l'espace des paramètres $\{a, b\}$ (ou espace de Hough). Chaque point (x, y) dans l'espace euclidien correspond à une droite $d'_{\{a,b\}} : b = y - ax$ dans l'espace de Hough (et vice-versa). La figure 3.2 montre que les droites $d'_{\{a,b\}}$ correspondant à tous les points d'une droite $d_{\{x,y\}}$ donnée se croisent en un seul point (a, b) dans l'espace des paramètres. Ce point (a, b) correspond aux paramètres de $d_{\{x,y\}}$.

Version polaire La transformée de Hough classique ne permet pas de représenter les droites verticales, car leur pente est infinie et elles ne croisent jamais l'axe des ordonnées. La figure 3.3 montre le résultat d'une transformée de Hough sur une telle droite : les droites $d'_{\{a,b\}}$ sont parallèles et se croisent à l'infini.
Pour remédier à ce problème, une version polaire de la transformée de Hough a été développée [12]. En coordonnées polaires, la droite est définie par son angle θ et sa distance de l'origine ρ : $y = (\frac{-\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta})$. L'équivalent dans l'espace de Hough est $r = x\cos\theta + y\sin\theta$. La figure 3.4 montre un exemple de transformée de Hough polaire.

3.2.1 Estimation de plans

Des extensions 3D de la transformée de Hough pour trouver des plans dans des nuages de points ont été proposées [52], puis en version polaire [5]. Cette extension au cas 3D se fait naturellement. Comme en 2D, et pour les mêmes raisons, il est ici aussi préférable d'utiliser la version polaire de l'algorithme. De plus, un plan en coordonnées polaires est décrit par seulement trois paramètres, un de moins qu'en coordonnées cartésiennes :

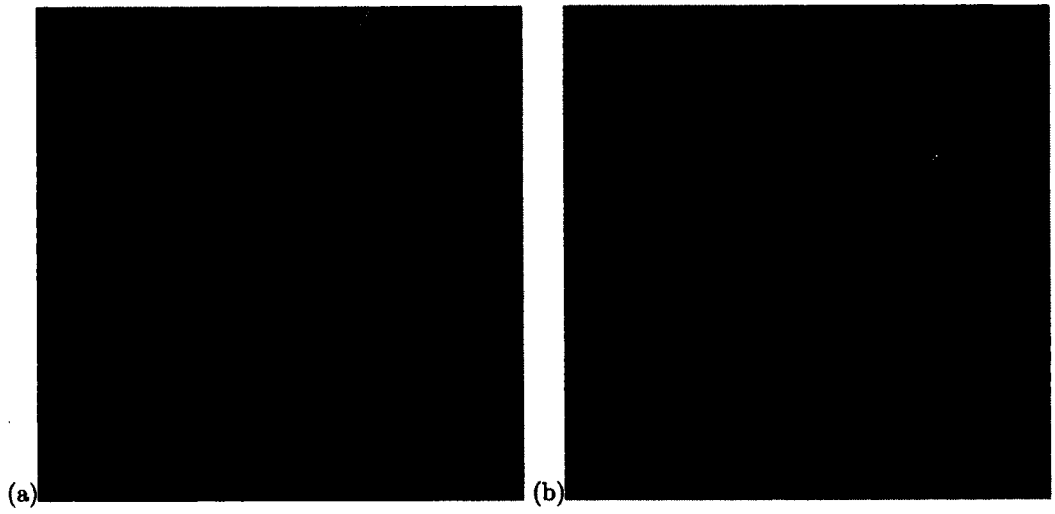


Figure 3.2 – Une droite dans l'espace euclidien (a) et les droites correspondant à ses points dans l'espace de Hough (b).

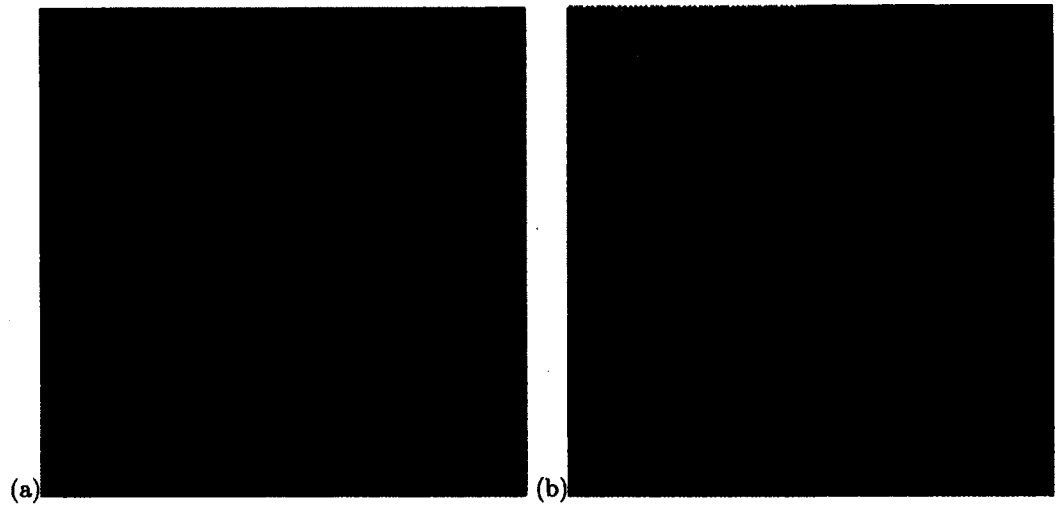


Figure 3.3 – Une droite verticale et sa transformée de Hough. Les paramètres de la droite ne peuvent pas être identifiés.

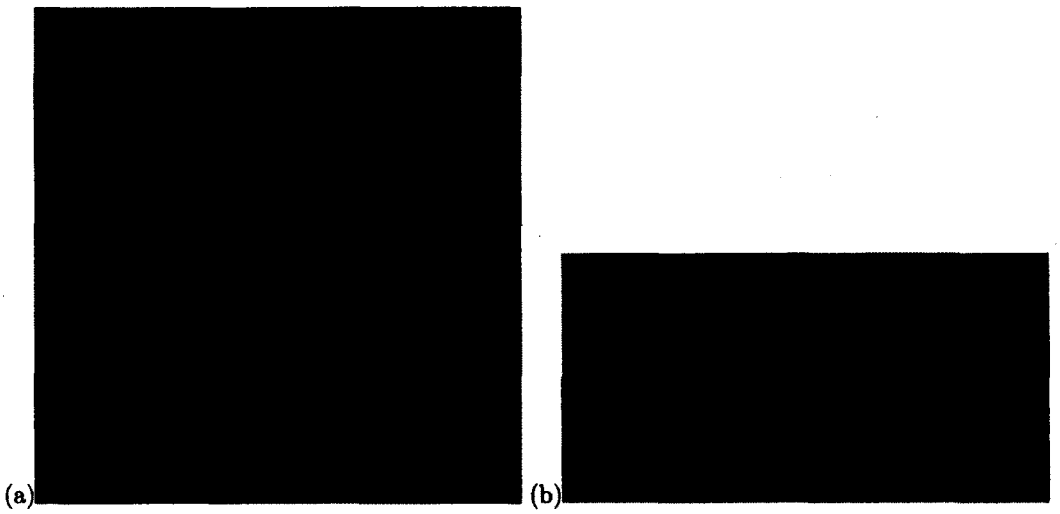


Figure 3.4 – Transformée de Hough polaire

- (θ, ϕ) : le vecteur normal en coordonnées sphériques ;
- ρ : la distance minimale du plan à l'origine,

où $\theta \in [0, \pi]$ et $\phi \in [-\pi/2, \pi/2]$. L'algorithme de la transformée de Hough 3D est décrit dans l'algorithme 4.

Algorithme 4 : Transformée de Hough 3D

Entrée : un nuage de points 3D N
Sortie : Π : l'équation du plan formé par N

pour tout $pt \in N$ **faire**
  pour $\phi \leftarrow -\pi/2$ **à** $\pi/2$ **faire**
    pour $\theta \leftarrow 0$ **à** 2π **faire**
      $\rho \leftarrow \cos(\phi)\cos(\theta)pt.X + \cos(\phi)\sin(\theta)pt.Y + \sin(\phi)pt.Z$
      $\mathsf{espParams}(\theta, \phi, \rho) \leftarrow \mathsf{espParams}((\theta, \phi, \rho) + 1$
    fin
  fin
fin
$\Pi \leftarrow argmax_{(\theta,\phi,\rho)}$ (espParams)

Temps d'exécution La transformée de Hough 3D a un temps d'exécution plutôt long, en comparaison avec son équivalent 2D. Cette différence s'explique par le nombre de cellules qui est considérablement plus élevé dans l'espace de Hough 3D que dans l'espace de Hough 2D. Une façon de réduire le temps d'exécution est d'utiliser un espace de Hough discrétisé plus grossièrement.

Affinage de la solution Afin d'éviter la perte de précision causée par une discrétisation plus grossière de l'espace, il est possible d'affiner la solution. Pour chaque plan $\hat{\pi}_i$ trouvé, un second espace de Hough est construit autour de la cellule correspondant à $\hat{\pi}_i$. Ce second espace est constitué de cellules plus fines et permet d'obtenir une estimation plus précise du plan π_i.

Estimation de plusieurs plans Si plus d'un plan doit être trouvé, n par exemple, l'algorithme retient plutôt les n cellules de l'espace des paramètres avec la plus grande valeur. Un problème qui survient généralement est que les cellules voisines d'un maximum ont elles aussi des valeurs élevées, davantage que les autres maxima correspondant aux autres plans. Une solution, utilisée entre autres par Matlab, est de réduire à zéro la valeur des cellules voisines d'un point d'accumulation. Toutefois, nous avons constaté qu'il est difficile de choisir une taille de voisinage appropriée. Nous avons donc opté pour une solution différente. Pour chaque plan π_i trouvé, nous décrémentons toutes les cellules de l'espace de Hough associées aux points concordants de π_i. Nous cherchons ensuite dans l'espace de Hough le nouveau point d'accumulation maximal, correspondant au plan π_{i+1}.

3.2.2 Estimation de quadriques

Une méthode de reconstruction d'arbres utilisant la transformée de Hough a été proposée (voir [40]), approximant les arbres par des cylindres. Un cylindre peut être représenté par cinq paramètres, mais les auteurs procèdent plutôt en deux étapes : une première transformée de Hough en 2D pour l'orientation, puis une seconde en 3D pour la position et le rayon. Également, les

créateurs de RANSAC en ont eux-mêmes présenté une application à la détection de cylindres (voir [18]).

Toutefois, étant donnés les résultats peu probants obtenus en estimant des plans avec la transformée de Hough (voir section 3.6), nous n'avons pas utilisé cette méthode pour estimer des quadriques. De plus, tel que mentionné précédemment, l'ajout d'une dimension à l'espace de Hough entraîne une augmentation considérable du temps d'exécution. L'espace de Hough à 10 dimensions qu'exige l'estimation de quadriques rend le temps d'exécution de l'algorithme prohibitivement long. L'estimation de sphères par une transformée de Hough, exigeant un espace à 4 dimensions, est également peu recommandée en raison de son temps d'exécution trop long.

3.3 Optimiseurs stochastiques : RANSAC

La régression est mathématiquement la méthode permettant d'estimer le plus précisément un modèle à partir de données bruitées (voir figure 3.5a). Toutefois, les régressions ne donnent pas de bons résultats en présence de données aberrantes (voir figure 3.5b). Pour pallier ce problème, des estimateurs robustes ont été développés. Ces estimateurs accordent moins d'importance aux valeurs extrêmes. Par exemple, il a été proposé de minimiser la somme des différences absolues [42] plutôt que la somme des différences au carré. Ainsi, les points aberrants ne voient pas leur influence amplifiée par la mise au carré. Mieux encore, Rousseeuw et Leroy [42] montrent que la méthode des moindres médianes (*Least Median of Squares*) permet de bien estimer le modèle même si l'ensemble de données contient jusqu'à 50 % de points aberrants. La méthode des moindres médianes trouve le modèle minimisant

$$med\left[(f(X_i) - \epsilon_i)^2\right].$$

Les auteurs montrent également que 50 % est la plus grande proportion de données aberrantes qu'une méthode de régression peut tolérer. Toutefois, les méthodes stochastiques[3], dont RANSAC est la plus connue, tolèrent des proportions de données aberrantes encore plus élevées que

3. Méthodes stochastiques : Méthodes basées sur ou utilisant le hasard.

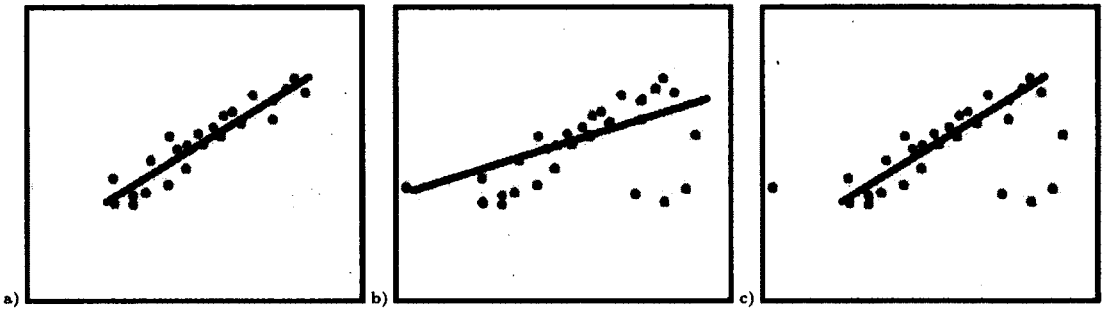


Figure 3.5 – L'estimation par les moindres carrés trouve le modèle minimisant la distance aux observations (a). Cette méthode est par contre très sensible aux données aberrantes (b). RANSAC, estimateur robuste, parvient à estimer correctement les paramètres de la droite, même en présence de données aberrantes (c).

50 %.

RANSAC (*RANdom SAmple Consensus*) est un algorithme d'estimation de modèle en présence de données aberrantes (voir [17]). Plutôt que d'estimer le modèle d'après l'ensemble des données S entier, incluant les données aberrantes, RANSAC choisit aléatoirement plusieurs sous-ensembles de données pour estimer le modèle. Pour chaque modèle estimé, RANSAC vérifie quelle proportion ρ de l'ensemble de données cadre avec ce modèle (moyennant une certaine marge d'erreur ϵ). Ces données sont appelées des points concordants (*inliers*) et les autres des points aberrants (*outliers*). Le modèle choisi est celui qui maximise ρ. Généralement, lorsque le nombre d'itérations dépasse un certain seuil sans avoir trouvé de modèle satisfaisant, on considère que l'estimation a échoué. Ce seuil dépend de la proportion de points concordants w et du nombre de dimensions s des données. Hartley et Zisserman [23] proposent de calculer le nombre d'itérations à effectuer ainsi :

$$N = \frac{\log(1-\psi)}{\log(1-w^s)}.$$

Suivant cette formule, il est certain avec une probabilité ψ qu'après N itérations de RANSAC, au moins un sous-ensemble de données exempt de points aberrants a été sélectionné. Hartley et Zisserman [23] et Fischler et Bolles [17] donnent plus de détails et discutent des autres paramètres de RANSAC.

3.3.1 Estimation de plans

Brenner [8] et Forlani et al. [19] décrivent des applications de RANSAC pour estimer des plans et reconstruire des bâtiments à partir d'un nuage de points. Une étude comparant la transformée de Hough et RANSAC dans le cadre de la reconstruction de bâtiments par plans a également été faite [50].
Dans le cadre de RANSAC, l'estimation de plans est simplement faite au moyen d'une régression linéaire (voir 3.1.1).

Points concordants et aberrants Le critère d'arrêt de RANSAC étant basé sur la quantité de points qui concordent avec le modèle estimé, il est important de savoir les distinguer des points aberrants. Cette segmentation se base sur la distance (absolue) de chaque point au plan estimé. Cette distance $d_\perp$, dite géométrique, est, dans le cas des plans, la même que la distance algébrique (voir équation 3.1). Si $d_\perp$ est inférieure à un seuil prédéterminé, le point considéré est un point concordant, sinon il est un point aberrant. Hartley et Zisserman [23] proposent une façon de calculer ce seuil. Il est préférable d'utiliser la distance géométrique, car le seuil a alors une réelle signification.

Estimation de plusieurs primitives À la suite de la mise en correspondance, le nombre de primitives n'est pas forcément connu et surtout, il n'est pas connu quel point appartient à quelle primitive. Nous avons donc étendu le RANSAC classique en une méthode itérative de segmentation (voir algorithme 5).

3.3.2 Estimation de quadriques

RANSAC peut également être utilisé pour segmenter les points 3D obtenus en plusieurs quadriques (ou sphères). On cite notamment Schnabel et al. [46], qui utilisent une méthode fondée sur RANSAC pour estimer des plans, des sphères, des cylindres, des cônes et des tores.

Algorithme 5 : Estimation de plusieurs primitives avec RANSAC

Entrée : S
Sortie : eq, les équations des primitives
nb, le nombre de primitives

```
1 tailleMin ← s *taille (S)
2 nb ← 0
3 répéter
4 |  {[eq [nb ], conc [nb ], aberr [nb ]} ← RANSAC (S)
5 |  nb ← nb + 1
6 |  S ← aberr
7 jusqu'à taille (conc [nb-1]) >= minsize
```

Dans notre cas, l'algorithme est exactement le même que pour les plans, à la différence que les sous-ensembles de points doivent contenir davantage de points (au moins neuf) pour estimer des quadriques. Pour estimer des sphères, trois points suffisent. Dans ces deux cas, les méthodes décrites respectivement aux sections 3.1.2 et 3.1.3 sont utilisées. Le critère utilisé pour séparer les points aberrants des points concordants est l'erreur de Sampson.

3.4 Nuées dynamiques

Nous avons quelque peu expérimenté avec une méthode d'estimation de plans basée sur l'algorithme des nuées dynamiques (*k-means* ; voir [35]).

Voisinages

La première étape de cette méthode est de décomposer le nuage de points (x_c, y, x_p) en z voisinages locaux composés de h points chacun. Ces voisinages doivent être choisis de façon à ce que les points soient le plus près possible de leur centre de gravité. La difficulté ici réside dans le fait que, après avoir constitué plusieurs voisinages, il devient possible de voir de petits groupes de points isolés être jumelés à d'autres points qui en sont relativement éloignés (et

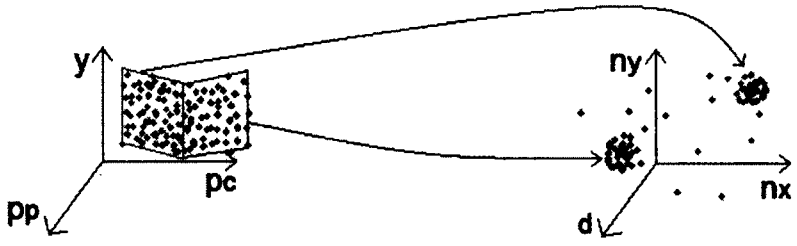


Figure 3.6 – (Gauche) Les points de l'espace projectif sont segmentés en sous-ensembles ou voisinages locaux. Pour chaque voisinage, un plan est estimé. (Droite) Un nuage de points dans l'espace (n_x, n_y, d) des paramètres du plan est ainsi obtenu. L'algorithme des nuées dynamiques est utilisé pour segmenter ce nuage en plans. Ici, une scène à deux plans (l'un en rouge, l'autre en vert) est présentée. Les points aberrants dans l'espace (n_x, n_y, d) correspondent aux voisinages situés près de la jonction des deux plans, contenant des points appartenant à l'un et à l'autre.

qui appartiennent potentiellement à une autre primitive) pour constituer un voisinage de taille suffisante, mais qui n'est plus vraiment local.

Pour constituer le premier voisinage, un point du nuage est choisi aléatoirement et les $h-1$ points qui en sont les plus près sont trouvés. Les points de ce nouveau voisinage sont retirés du nuage de points. Cette procédure est répétée jusqu'à ce que le nuage contienne moins de h points.

Une fois tous les voisinages constitués, un plan doit être estimé pour chacun. Le plan qui minimise la distance (euclidienne) aux points de chaque voisinage est trouvé en utilisant une régression linéaire. z plans sont obtenus, chacun étant défini par son vecteur normal et sa distance de l'origine (perpendiculairement). La composante n_z du vecteur normal étant fixée à 1 (voir section 3.1.1, chaque plan correspond à un point dans un espace (n_x, n_y, d). La figure 3.6 illustre l'estimation des plans locaux.

L'algorithme des nuées dynamiques est enfin utilisé pour trouver l'équation des n plans qui constituent le nuage de points à partir des z plans trouvés à l'étape précédente. L'algorithme des nuées dynamiques (voir algorithme 6) vise à séparer les données en k groupes dont l'inertie par rapport au centre de gravité est faible. Comme l'indique MacQueen [35], il n'est pas garanti que la configuration optimale soit trouvée, mais l'algorithme des nuées dynamiques s'en approche

généralement bien. Le nombre de groupes k (comme le nombre de dimensions des données du problème) doit être déterminé au préalable, faute de quoi le problème est NP-complet.

Algorithme 6 : Nuées dynamiques

initialiser les groupes (aléatoirement ou autrement)
répéter
  calculer le centre de gravité de chaque groupe
  associer chaque point au groupe dont le centre de gravité en est le plus près
jusqu'à *composition des groupes ne change plus*

3.5 Espace d'estimation

Les points obtenus par la mise en correspondance sont dans un espace dit projectif, dont les axes sont les coordonnées x et y de la caméra et la coordonnée x du projecteur, dénotés $\{x_c, y, x_p\}$. Dans cet espace, il est certain que les composantes x_c et y ne sont jamais bruitées, car elles correspondent exactement à des pixels de la caméra. Seule la composante x_p peut être bruitée, dans le cas d'une mauvaise mise en correspondance. Comme l'explique l'annexe C, les points peuvent être convertis vers l'espace euclidien, le monde $\{X, Y, Z\}$ habituel. Toutefois, cette conversion entraîne une propagation du bruit en x_p vers les trois composantes X, Y, Z. Pour cette raison, il est préférable de faire l'estimation des primitives dans l'espace projectif, puis de convertir le résultat vers l'espace euclidien, plutôt que l'inverse. Alors, la distance géométrique à utiliser pour discriminer les points concordants et aberrants est la distance sur l'axe x_p, et non la distance euclidienne classique[4]. Des mesures quantitatives justifiant ces conclusions sont présentées à la section 4.3.6.

On note qu'il est possible d'estimer tant les plans que les quadriques dans l'espace projectif. Toutefois, on n'utilise pas directement une distance géométrique dans l'estimation de quadriques. Il est détaillé en 3.1.2 comment est faite l'estimation de quadriques.

4. On conserve le nom de distance euclidienne pour la mesure $\sqrt{\sum_{i=i}^{n}(x_i - x_i')^2}$, même si les points ne sont pas dans l'espace euclidien.

	Nuées dynamiques			Hough			RANSAC		
	Points aberrants			Points aberrants			Points aberrants		
	0	25 000	50 000	0	25 000	50 000	0	25 000	50 000
0	63,33	4,58	0,83	37,5	37,92	37,92	100	99,58	83,75
σ 5	28,75	0,83	0	32,5	33,75	32,92	92,5	80,42	63,33
10	7,08	0,42	0	25,42	23,75	27,5	84,17	76,67	61,25

Tableau 3.1 – Pourcentage de plans correctement retrouvés, en moyenne, en fonction de la méthode d'estimation, du bruit gaussien et du nombre de points aberrants ajoutés.

3.6 Étude comparative sur l'estimation de plans

Afin de choisir la meilleure méthode d'estimation de plans, nous avons réalisé des tests sur des données synthétiques. Les facteurs les plus importants dans le choix de la méthode sont la capacité d'estimer plusieurs primitives et la résistance au bruit et aux données aberrantes. En effet, les points 3D obtenus par l'étape de mise en correspondance sont souvent perturbés par des erreurs de ces types. Pour simuler ces conditions, nous avons généré aléatoirement des scènes contenant des points distribués sur huit plans dans un volume de 5000 × 5000 × 3000. Chaque plan est compososé de 5000 points, pour un total de 40 000 points. Ces points sont ensuite corrompus par un bruit gaussien d'écart-type $\sigma^2 \in \{0, 5, 10\}$. De plus, des points aberrants sont ajoutés à la scène. Pour chacune de ces 9 configurations, nous avons généré 30 scènes sur lesquelles nous avons lancé les algorithmes de RANSAC, de la transformée de Hough et des nuées dynamiques. Ces algorithmes sont décrits respectivement aux sections 3.3.1, 3.2.1 et 3.4. RANSAC s'est nettement démarqué. En effet, même dans le pire cas (50 000 points aberrants et un bruit gaussien d'écart-type 10), RANSAC parvient à retrouver en moyenne 5 des 8 plans (voir tableau 3.6).

Nous avons également comparé les trois mêmes méthodes avec des données issues d'une projection dans notre environnement virtuel. Nous avons projeté 30 images de la séquence *live band* (voir section 4.2) avec un bruit gaussien d'écart-type 8 et une réduction de contraste de 33%.

Algo.	Plans	r (%)	$\angle$	$\delta_d(\%)$
Nuées dyn,	1	0	-	-
Hough	1	100	1,8987	0.67
RANSAC	1	100	0,1675	0.71
Nuées dyn,	4	0	-	-
Hough	4	25	-	-
RANSAC	4	92,74	0,6774	1.27

Tableau 3.2 – Pourcentage de succès (r) et mesures d'erreur sur l'angle et la distance à l'origine (δ_d) pour l'estimation de plans par nuées dynamiques, transformée de Hough et RANSAC. Des scènes à un et quatre plans ont été utilisés avec une projection de 45 images de la séquence *live band*.

Ces valeurs ont été choisies arbitrairement et visent à simuler l'effet d'une vraie caméra avec un court temps d'exposition. Nous avons généré les cartes de disparité et points 3D pour des scènes à un et quatre plans, puis lancé les algorithmes d'estimation. Les résultats obtenus figurent dans le tableau 3.2. On constate que dans la scène à quatre plans, RANSAC est la seule méthode qui parvient à correctement estimer les équations des quatres plans. Les résultats obtenus avec la scène à un plan montrent en outre que la précision de la transformée de Hough est limitée par la discrétisation de l'espace utilisée.

Pour mettre RANSAC davantage à l'épreuve, nous l'avons également testé sur la carte de disparité de la séquence *Venus*, de l'ensemble de données de *Middlebury College* [45]. La figure 3.7 présente la carte de disparité étalon de cette scène, constituée de quatre plans, à laquelle nous avons ajouté un fort bruit gaussien. Même dans ces conditions difficiles, RANSAC parvient à retrouver les équations des quatre plans avec moins de 3 degrés d'erreur.

Sphères Pour valider l'efficacité de RANSAC sur d'autres types de primitives que les plans, nous l'avons utilisé pour estimer des sphères générées aléatoirement. Nous avons généré aléatoirement des points formant 8 sphères dans un volume de $5000 \times 5000 \times 3000$. Chaque point a été corrompu par un bruit gaussien d'écart-type $\sigma^2 \in \{0, 5, 10\}$. Des points aberrants (0, 25 000, 50 000) ont également été ajouté à la scène. Pour chaque configuration, nous avons lancé l'algorithme d'estimation de sphères (voir 3.1.2) sur 30 scènes différentes. Les résultats obtenus sont présentés dans le tableau 3.6.

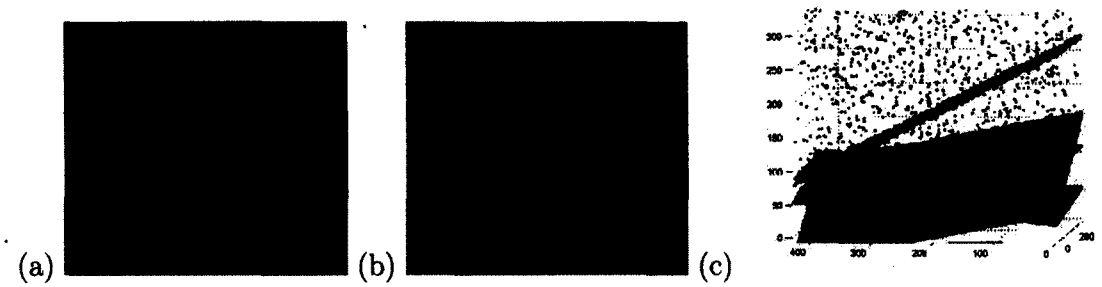


Figure 3.7 – (a) Carte de disparité étalon de la séquence *Venus* (voir [45]) (b) où 25% des pixels sont corrompus par un bruit gaussien. (c) Vue 3D des points reconstruits. Les plans retrouvés sont en vert et les points concordants en rouge. Les points aberrants sont en noir. On note que la plupart de ceux-ci se situent hors du cadre de cette scène.

	RANSAC		
	Points aberrants		
	0	25 000	50 000
0	0.9833	0.5167	0.1792
5	0.8292	0.3167	0.1458
10	0.7542	0.3167	0.1833

Tableau 3.3 – Pourcentage de sphères correctement retrouvées, en moyenne, en fonction du bruit gaussien et du nombre de points aberrants ajoutés. L'algorithme d'optimisation utilisé est RANSAC (voir 3.1.2).

3.7 Conclusion

Nos expérimentations et les résultats obtenus montrent que l'algorithme RANSAC est celui qui répond le mieux à nos besoins. Il permet de bien estimer les primitives même en présence de fort bruit ou de points aberrants. L'algorithme des nuées dynamiques fonctionne en général bien dans les cas simples et idéaux, mais il n'est guère robuste au bruit. Dans les cas bruités, il ne parvient jamais à bien retrouver tous les plans. Enfin, la transformée de Hough permet difficilement de retrouver un nombre exact et plus grand que 3 de primitives dans un nuage de points. De plus, la transformée de Hough a une précision limitée par le niveau de discrétisation utilisé.

CHAPITRE 4

Expérimentation et résultats

Afin de valider l'efficacité de notre méthode et de la comparer avec d'autres méthodes fréquemment utilisées, nous avons réalisé plusieurs tests et expérimentations. Leurs résultats sont consignés ici. Certaines de ces expérimentations ont été réalisées dans le cadre d'une collaboration avec MM. Marc-Antoine Drouin du CNRC et Pierre-Marc Jodoin de l'Université de Sherbrooke. Les résultats de ces travaux apparaissent dans l'article [11]. De plus, certaines de ces expérimentations ont été réalisées dans un environnement virtuel spécialement conçu par l'auteur (voir annexe D).

4.1 Cartes de disparité

Les cartes de disparité présentées dans ce mémoire ont été obtenues dans notre environnement virtuel. Ainsi, il nous est possible de générer pour chaque scène une carte de disparité étalon. La figure 4.1 présente ces étalons.

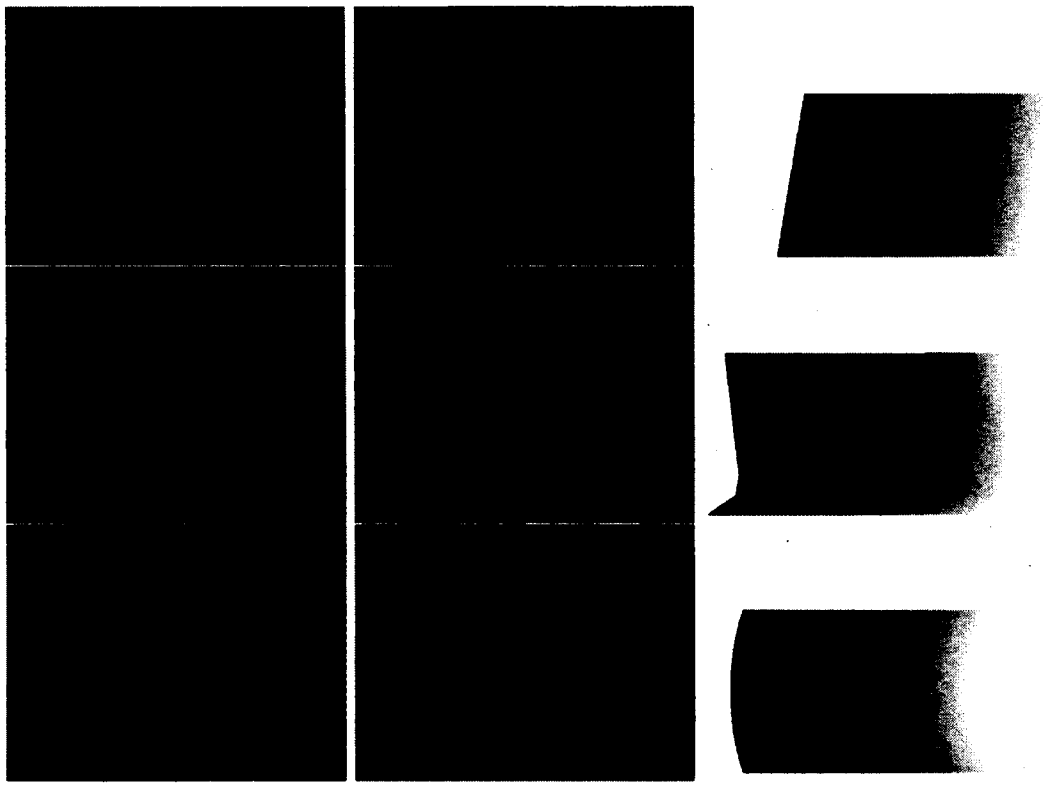


Figure 4.1 – (Gauche) Image de la scène vue d'un point de vue externe, (centre) vue de la caméra et (droite) carte de disparité étalon pour des scènes à un et quatre plans et à deux sphères. On note que la caméra (en rouge) et le projecteur (en vert) sont visibles dans les images du point de vue externe. Le cadre noir délimite la région de la scène où la projection est faite.

4.2 Notre méthode

Nous avons mis notre méthode à l'épreuve en l'utilisant pour reconstruire des scènes composées d'un, deux et quatre plans, ainsi que de scènes composées de quadriques. Nous avons utilisé trois séquences vidéo différentes (voir figure 4.2). La première, nommée *corridor* (CRD), filmée par une caméra fixe, montre un homme marchant de gauche à droite. La seconde, *live band* (LB), est un extrait d'un concert filmé avec un téléphone cellulaire par une personne dans la foule. Non seulement la caméra y est instable, mais les images présentent des artefacts de compression et beaucoup de mouvement. La troisième séquence, *traffic circle* (TC), montre des voitures traversant un carrefour giratoire. Dans cette séquence, la caméra se déplace de gauche à droite. Ce type de mouvement de la caméra est fréquent dans les vidéos de la vie quotidienne ou d'événements privés. Les séquences contiennent respectivement 167, 240 et 1036 images et ont été filmées (respectivement) avec une caméra professionnelle, un téléphone cellulaire et un appareil photo numérique de milieu de gamme.

Dans tous les cas, les paramètres utilisés par notre méthode sont les suivants :

- $\alpha = 0.85$
- $L = 6$
- $COSTMAX = 2$
- $s = 0.1$

Les résultats identifiés comme les nôtres (LNS, lumière non structurée) ont été obtenus avec la méthode de détection de mouvement avec un seuil adaptatif et l'estimation de primitives 3D dans l'espace projectif.

Deux projecteurs ont aussi été utilisés. Le premier, utilisé avec la séquence *corridor* a une résolution de 1576 × 1080. Le second, utilisé avec la séquence *live band*, a une résolution de 1024 × 768 et fonctionne avec des DEL de 225 lumens. Dans les deux cas, nous avons utilisé une caméra de 3024 × 4334 dont les images ont été redimensionnées pour être de la même taille que celles des vidéos projetées.

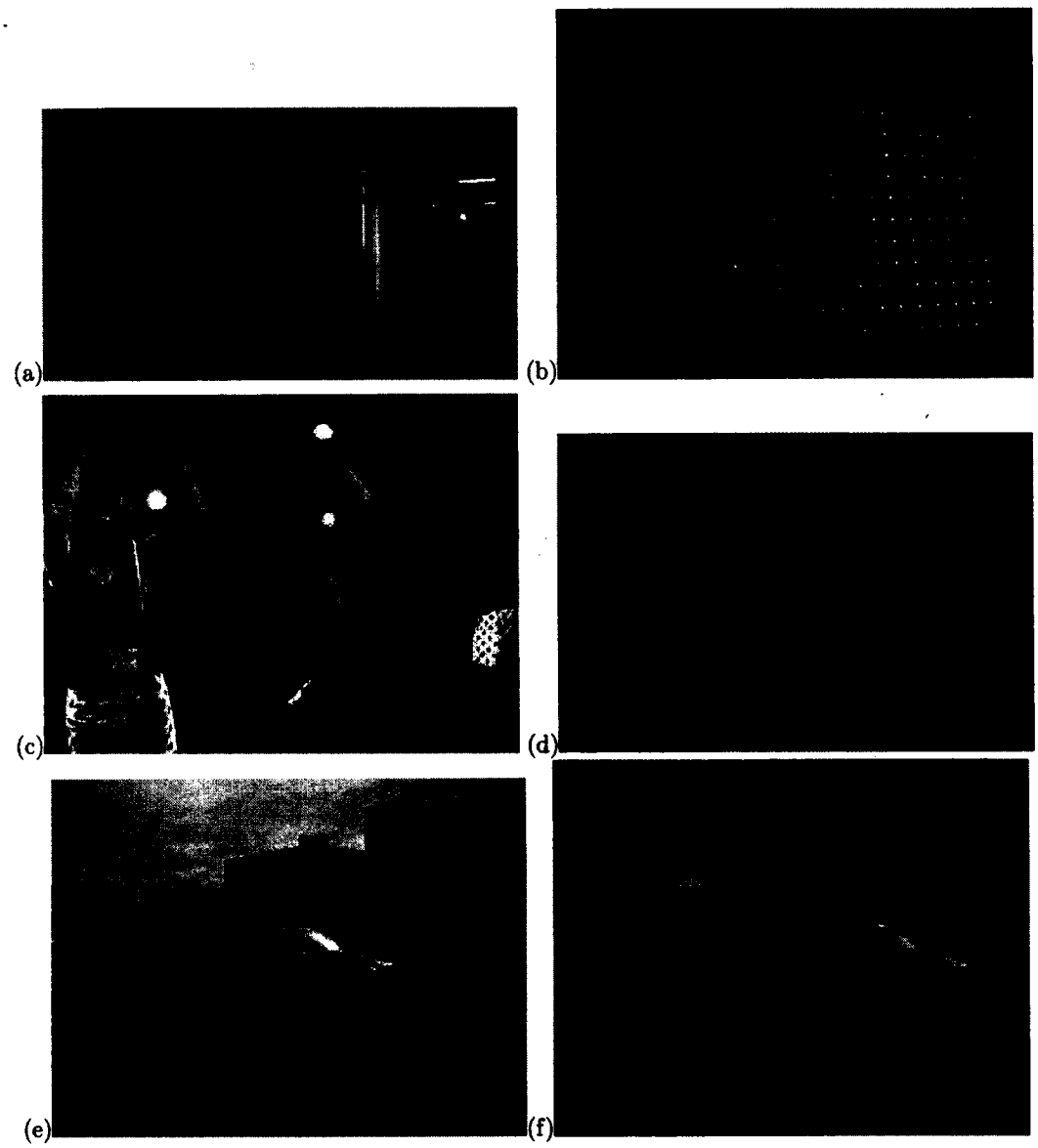


Figure 4.2 – Les séquences *corridor*, *live band* et *traffic circle* utilisées dans nos tests. À gauche, l'image projetée et à droite, l'image acquise par la caméra. Le faible contraste dans les images (b) et (d) s'explique par le court temps d'exposition de la caméra, fixé à 1/30e de seconde, pour assurer la synchronisation avec le projecteur. (L'image (f) ne souffre pas de cette dégradation, ayant été acquise dans notre environnement virtuel.)

4.3 Comparaison de différentes fonctions de coût

Nous avons constaté que la distance euclidienne entre les vecteurs de mouvement ne donnait pas des résultats suffisamment bons pour nos besoins (voir section 2.3.3). Tel qu'expliqué à la section 2.3.3, nous avons créé notre propre fonction de coût (voir équation 2.4). Afin de valider son efficacité, nous avons exécuté l'algorithme de mise en correspondance en utilisant la distance euclidienne, la distance de Manhattan et notre fonction de coût. Pour constituer notre banc de tests, nous avons réalisé des projections sous différentes conditions de bruit σ et de contraste γ dans notre environnement virtuel. Ces deux dégradations sont appliquées aux images de la caméra. σ est l'écart-type du bruit gaussien, alors que γ est le pourcentage de dégradation de contraste. Deux scènes, deux séquences et une fenêtre temporelle de $W = 30$ ont été utilisées. La mesure d'efficacité utilisée est la distance moyenne entre les points reconstruits et la primitive originale à laquelle chacun appartient. RANSAC a été utilisé pour associer chaque point à la bonne primitive. Les résultats obtenus sont présentés au tableau 4.1. Avec la distance euclidienne entre les vecteurs de mouvement, la distance moyenne obtenue augmente rapidement lorsque les conditions de projection se dégradent. La distance de Manhattan fonctionne bien, mais notre fonction de coût (*quanta*) donne de meilleurs résultats. Il est à noter que les cas où toutes les primitives n'ont pas été retrouvées correctement ont été ignorés et sont indiqués par un tiret dans le tableau 4.1.

4.3.1 Comparaison avec SIFT

Afin de valider l'efficacité de notre algorithme de mise en correspondance, nous l'avons comparé avec SIFT (voir section 1.3.3). Nous avons pour ce faire remplacé dans notre méthode l'algorithme de mise en correspondance par SIFT. Puisque notre méthode utilise plusieurs paires d'images et SIFT une seule, nous avons cumulé les correspondances obtenues par SIFT à chaque instant t.

Nous avons projeté la séquence *corridor* sur une surface plane texturée (voir section 4.3.2 et

		euclidienne				manhattan				quanta			
		γ (%)				γ (%)				γ (%)			
		0	30	60	90	0	30	60	90	0	30	60	90
1 plan	0	285.231	289.207	-	-	282.123	282.868	281.625	280.304	281.117	281.611	281.583	281.082
	5	286.675	287.875	-	-	282.307	282.855	283.119	277.86	281.035	282.039	281.238	278.413
	10	285.583	-	-	-	280.576	276.578	-	-	274.988	274.59	-	-
	20	291.636	-	-	-	274.633	-	-	-	274.627	-	-	-
4 plans	0	302.702	243.249	276.73	-	297.86	319.235	353.828	327.995	248.34	261.462	143.152	304.839
	5	148.618	230.916	184.999	-	219.106	316.033	303.111	228.459	88.9915	252.471	160.692	18.7892
	10	116.646	344.51	-	-	360.514	281.45	269.979	-	141.357	274.039	233.339	-
	20	249.471	123.221	-	-	278.418	345.98	-	-	271.727	133.443	-	-

Tableau 4.1 – Distance moyenne des points reconstruits aux primitives correspondantes pour différentes fonctions de coût. Une pénalité correspondant au nombre de points aberrants est appliquée lorsque les primitives ne sont pas toutes bien retrouvées. Ici, la fenêtre temporelle $W = 30$ images. σ est l'écart-type du bruit gaussien et γ le pourcentage de dégradation de contraste appliqués aux images de la caméra. Dans l'ensuemble, notre fonction de coût (*quanta*) donne les plus faibles mesures d'erreur.

figure 4.2). La figure 4.3 montre que notre méthode produit une erreur angulaire moyenne bien inférieure à celle produite par SIFT. La présence de marqueurs sur la surface et les fortes distorsions spatiale et photométrique réduisent SIFT à ne trouver que très peu de bonnes correspondances (moins de 50, voir figure 4.4). Ces piètres résultats causent une erreur de reprojection trois fois plus grande que celle obtenue avec notre méthode.

Nous avons également fait des tests pour voir comment notre méthode se compare à SIFT lorsque la surface de projection et inclinée à différents angles. Ces tests ont été réalisés dans notre environnement virtuel. Pour simuler l'effet assombrissant d'une caméra utilisant un temps d'exposition réduit (ce qui est nécessaire pour la synchronisation avec le projecteur), nous avons réduit le contraste des images de la caméra par un facteur variant de 1 à 4. Le tableau 4.2 présente le nombre de points concordants obtenus dans chaque configuration. Ces résultats montrent clairement que notre méthode trouve davantage de correspondances que SIFT, environ 1800 fois plus. Les correspondances trouvées par notre méthode sont également plus stables, particulièrement en présence de fortes dégradations de contraste. En effet, la variation entre les quantités minimale et maximale de correspondances est de $26,8\%$ pour notre méthode et de 99% pour SIFT. Les résultats du tableau 4.2 mettent aussi en lumière l'importance du seuillage

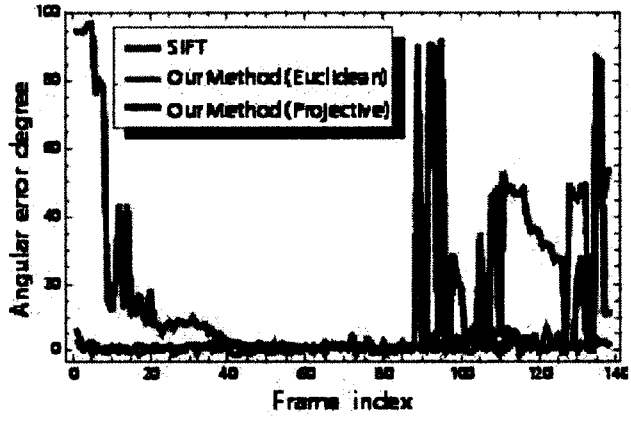


Figure 4.3 – Erreur angulaire entre la mesure étalon et le plan estimé par notre méthode et par SIFT. La séquence *corridor* a été utilisée.

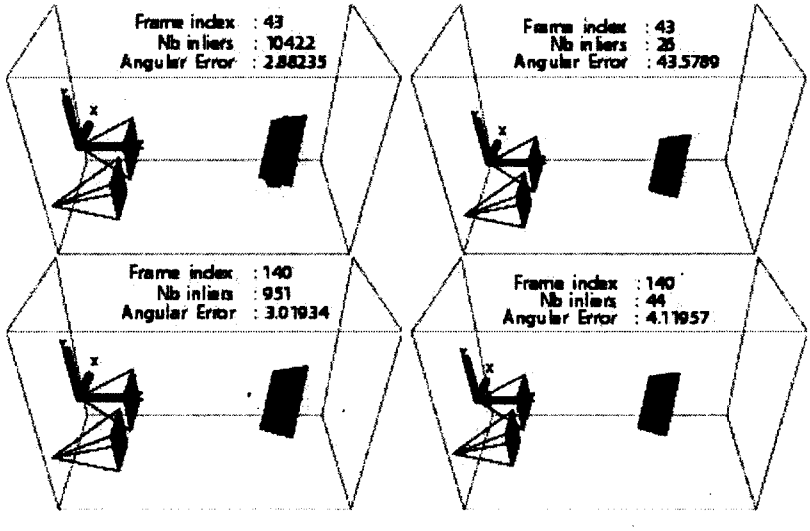


Figure 4.4 – Les points 3D retrouvés par notre méthode (gauche) et SIFT (droite) aux instants $t = 43$ et $t = 140$ en utilisant la séquence *corridor*. Comme le montre la figure 4.3, SIFT retrouve peu de points en $t = 43$, causant une erreur de plus de 40 degrés.

Séquence \ Angle	0^o	5^o	25^o	50^o
LNS (S.F.) - *LB - 1*	66373	62873	63845	43581
LNS (S.F.) - *LB - 2*	19897	20571	18941	17102
LNS (S.F.) - *LB - 4*	3414	3352	3192	2735
LNS (S.F.) - *CRD - 1*	62186	59763	57393	52375
LNS (S.F.) - *CRD - 2*	6071	6309	5910	5109
LNS (S.F.) - *CRD - 4*	824	823	670	637
LNS (S.A.) - *LB - 1*	50232	46670	35886	51035
LNS (S.A.) - *LB - 2*	49483	41300	36457	43952
LNS (S.A.) - *LB - 4*	46466	41900	34402	44228
LNS (S.A.) - *CRD - 1*	49722	45108	37291	49393
LNS (S.A.) - *CRD - 2*	48342	43086	38618	46358
LNS (S.A.) - *CRD - 4*	45800	42322	36997	46417
SIFT - *LB - 1*	2664	2897	3198	2269
SIFT - *LB - 2*	2117	2343	2612	1872
SIFT - *LB - 4*	1184	1358	1538	1055
SIFT - *CRD - 1*	276	337	398	331
SIFT - *CRD - 2*	169	213	247	224
SIFT - *CRD - 4*	25	38	49	47

Tableau 4.2 – Le nombre de points concordants trouvés par SIFT et par notre méthode (LNS) utilisant un seuil adaptatif (S.A.) et un seuil fixe (S.F.) lors de la détection de mouvement. Ces résultats ont été obtenus en projetant les séquences *corridor* (CRD) et *live band* (LB) dans notre environnement virtuel. Le nombre associé au nom des séquences est le facteur de dégradation de contraste appliqué aux images de la caméra. L'expérience a été faite sur un plan incliné à quatre angles différents par rapport à la caméra.

adaptatif, car le nombre de points concordants obtenus avec un seuil fixe chute drastiquement quand les images ont un faible contraste.

Enfin, nous avons projeté 100 images de la séquence *live band* sur la scène à quatre plans dans notre environnement virtuel, et utilisé SIFT pour faire la mise en correspondance. Les cartes de disparité résultant de ces tests sont présentées à la figure 4.5. Ces cartes de disparité sont nettement moins denses que celles obtenues avec notre méthode.

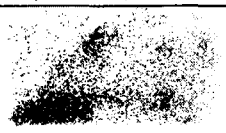
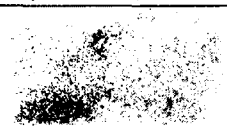
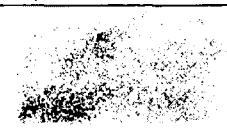
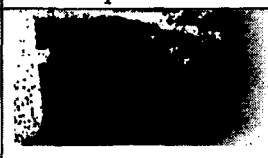
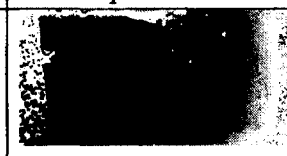
	$\gamma = 0\%, \sigma = 0$	$\gamma = 30\%, \sigma = 0$	$\gamma = 60\%, \sigma = 0$	$\gamma = 90\%, \sigma = 0$
a)	 $q = 12303$	 $q = 10808$	 $q = 7418$	$q = 0$
b)	 $q = 83901$	 $q = 82988$	 $q = 81550$	 $q = 79070$
	$\gamma = 0\%, \sigma = 10$	$\gamma = 0\%, \sigma = 20$	$\gamma = 0\%, \sigma = 30$	$\gamma = 0\%, \sigma = 40$
c)	 $q = 11461$	 $q = 9233$	 $q = 7224$	 $q = 5664$
d)	 $q = 83239$	 $q = 83004$	 $q = 80478$	 $q = 81452$

Figure 4.5 – Cartes de disparité obtenues avec SIFT et avec notre méthode. q est le nombre de points concordants obtenus. La rangée (a) montre que SIFT tolère très mal les fortes dégradations de contraste (γ). En fait, avec $\gamma = 90\%$, SIFT ne parvient à reconstruire qu'un seul point 3D. La rangée (c), en revanche, montre que SIFT parvient à reconstruire des points même en présence d'un fort bruit. Les rangées (b) et (d) présentent les résultats obtenus avec notre méthode.

4.3.2 Configuration à un plan

Cette configuration correspond au cas idéal où la scène est monoplane (un mur ou un écran, par exemple). Dans notre cas, la scène est en fait une cible de calibration plane, contenant des marqueurs à des positions connues. Nous avons ainsi pu déterminer l'équation du plan avec une méthode photogrammétrique et en faire notre mesure étalon. En outre, les marqueurs sont visibles dans la séquence filmée ; cette configuration permet donc de tester notre méthode sur une surface de projection texturée.

Nous avons d'abord testé notre méthode avec la séquence *corridor*. Puisque notre méthode se base sur le mouvement contenu dans la séquence, *corridor* est celle présentant le plus grand défi, en raison du peu d'activité qu'elle contient. Avec une fenêtre temporelle W de 30 images, nous avons estimé l'équation du plan à chaque instant t. Tel que présenté à la figure 4.3, nous avons obtenu une erreur moyenne de 2,8 degrés entre le plan estimé et la mesure étalon. La figure 4.3 montre également que l'estimation est plus robuste lorsque faite dans l'espace projectif (plutôt que dans l'espace euclidien). On y voit qu'au temps $t = 120$, tant l'estimation dans l'espace euclidien que l'estimation d'après les correspondances de SIFT échouent. La figure 4.6 montre l'image de la caméra au temps $t = 120$ et l'image de mouvement associée. Celle-ci ne contient que très peu de pixels actifs, environ $3,8\%$, mais notre méthode parvient néanmoins à bien estimer l'équation du plan.

De plus, comme le montre la figure 4.7, l'erreur obtenue de 2,8 degrés engendre une erreur de reprojection de moins de quatre pixels, pratiquement imperceptible à l'oeil nu. C'est le cas à la figure 4.8, où un damier déformé par notre méthode, en (a), est projeté sur une surface plane, en (b).

Enfin, nous avons également utilisé la séquence *live band* dans une projection monoplane. En utilisant une fenêtre temporelle W de 30, l'équation du plan a été estimée avec une erreur de moins de 3 degrés et en moyenne 13 545 points concordants ont été retrouvés (voir figure 4.4).

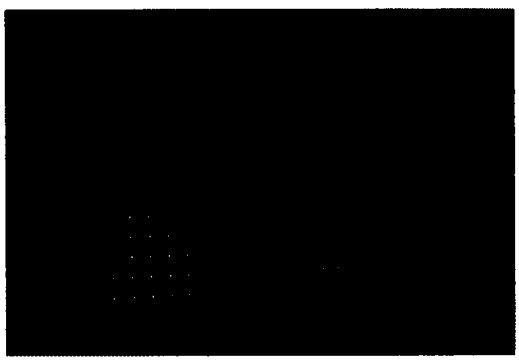


Figure 4.6 – Image de la caméra pour la séquence *corridor* à $t = 120$, contenant peu d'activité (les pixels actifs sont en noir dans l'image de droite).

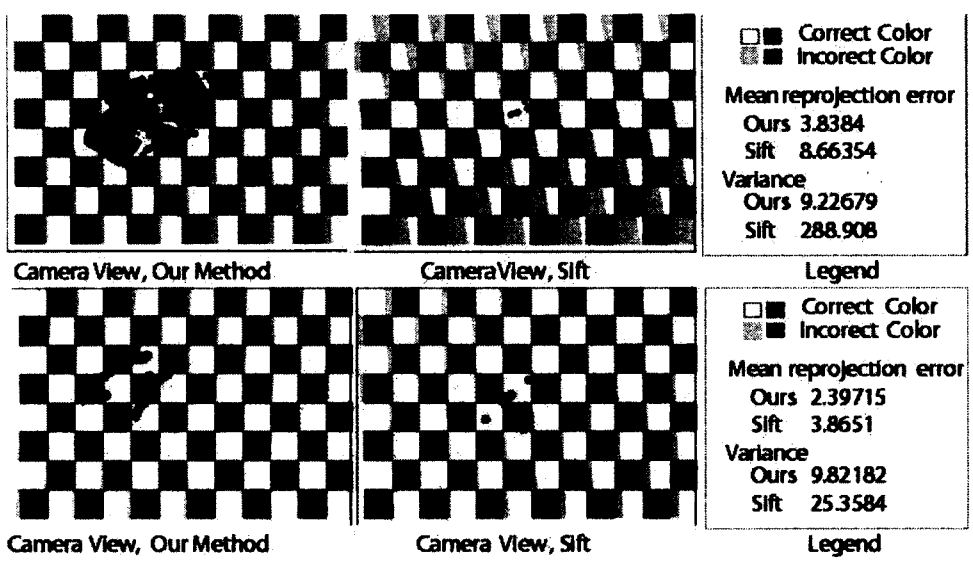


Figure 4.7 – Erreur de reprojection d'un damier pour la séquence *corridor*. Les points rouges sont les points 3D retrouvés par notre méthode (gauche) et par SIFT (centre). Les résultats pour deux instants t sont présentés, $t = 43$ et $t = 140$.

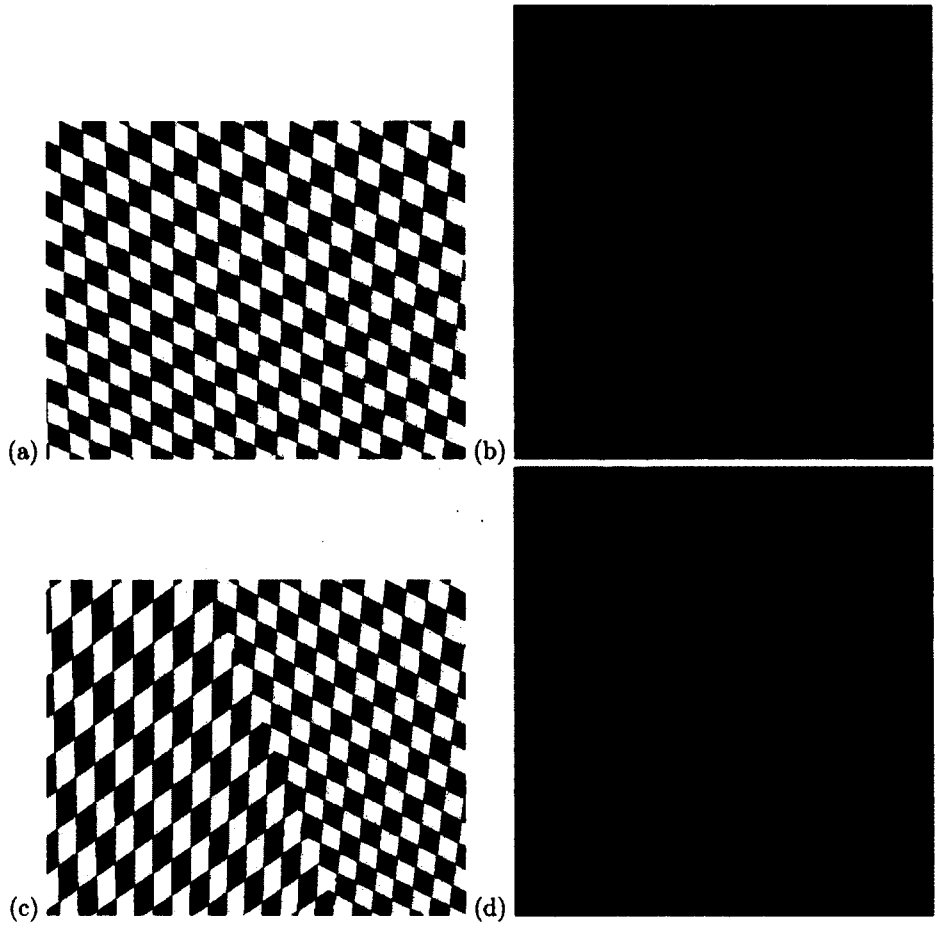


Figure 4.8 – (a),(c) Images déformées selon la scène 3D telle que reconstruite par notre méthode (un et deux plans). (b),(d) présentent la projection sur la scène des images déformées. L'erreur de reprojection de quatre pixels est pratiquement imperceptible.

4.3.3 Configuration à deux plans

Dans cette configuration, nous avons projeté la séquence *live band* sur une scène composée de deux plans s'entrecoupant, placés à 800 millimètres de la caméra. En guise de base de comparaison, nous avons utilisé une reconstruction de la scène obtenue par lumière structurée, en utilisant des codes de Gray et un décalage de phase (voir [44]). Nous avons effectué la même reconstruction avec notre méthode, avec des fenêtres temporelles W de 15, 30 (avec un seuil adaptatif) et de 60 (avec un seuil fixe). La figure 4.9 présente l'erreur angulaire obtenue pour chaque plan dans chacune des configurations. Elle présente également la superposition des reconstructions 3D de la scène, obtenues par lumière structurée et par notre méthode ($W = 30$, seuil adaptatif). L'erreur de reconstruction ne dépasse pas 4 millimètres, ou 0,5 %. L'erreur moyenne de reconstruction pour les deux plans est de 1,9 et 1,3 millimètres, tandis que l'erreur angulaire moyenne est de 1,5 degré avec $W = 15$ et de 0,9 degré avec $W = 30$. En comparaison, un seuil fixe nécessite un W plus grand pour obtenir des résultats similaires, mais engendre alors tout de même une plus grande variation entre les erreurs moyennes des deux plans. La figure 4.8 montre un damier déformé par notre méthode (c) projeté sur la scène biplane (d). Comme en 4.3.2, l'erreur de reprojection passe inaperçue.

Enfin, nous avons comparé le seuillage adaptatif à l'utilisation d'un seuil fixe avec des W variant de 5 à 60 images. Les résultats, présentés au tableau 4.3, concordent avec ceux de la figure 4.9 en montrant que le seuillage adaptatif est non seulement plus précis, mais requiert un W plus petit. Notamment, une fenêtre temporelle de seulement 25 images (moins d'une seconde) permet d'obtenir pour les deux plans une erreur angulaire moyenne de moins d'un degré.

4.3.4 Configuration à quatre plans

Nous avons projeté, dans notre environnement virtuel, la séquence *traffic circle* sur une scène composée de quatre plans. La figure 4.10 présente une esquisse de la scène (a), les points concordants retrouvés (b) et les plans estimés avec RANSAC dans l'espace projectif (c). L'erreur de

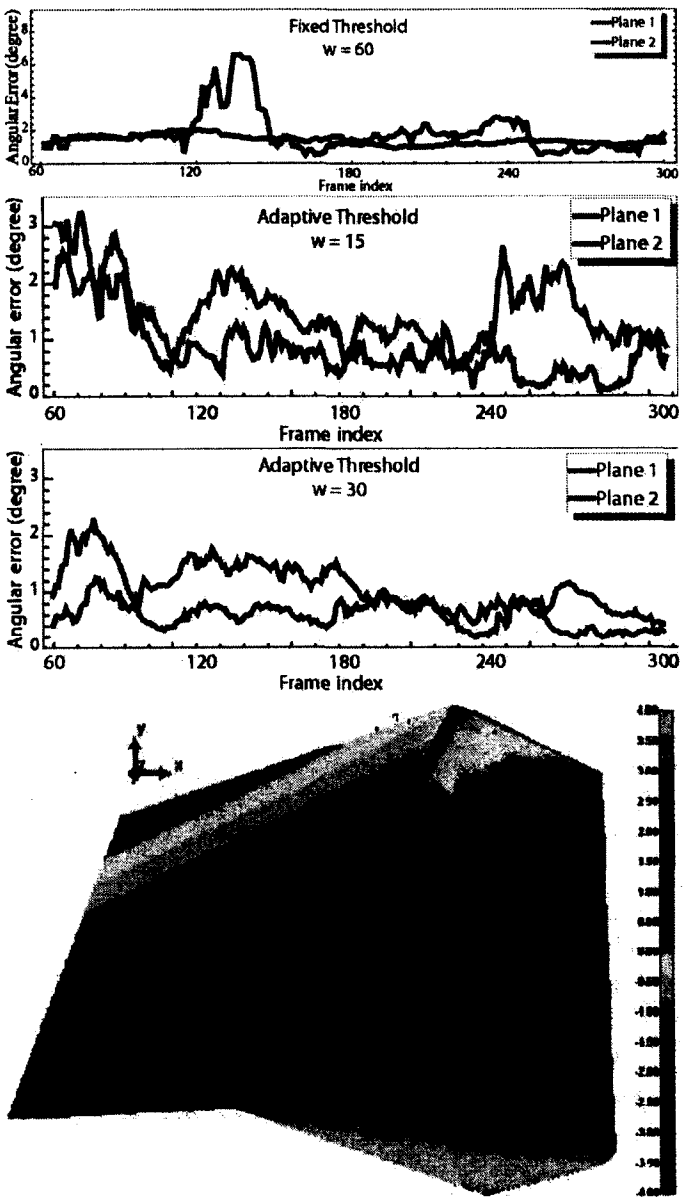


Figure 4.9 – (Haut) Erreur angulaire à chaque instant t pour deux plans, en utilisant un seuil fixe ou un seuil adaptatif. (Bas) La différence en millimètres entre le modèle 3D obtenu par lumière structurée (avec 14 patrons) et notre méthode ($W = 30$, S.A.).

	Séquence	Moyenne	Médiane	Variance	Maximum
Plan 1	$W = 60$ (S.F.)	1,22	1,18	0,08	1,88
	$W = 30$ (S.A.)	0,86	0,77	0,12	1,64
	$W = 25$ (S.A.)	0,95	0,83	0,17	1,98
	$W = 20$ (S.A.)	1,27	1,14	0,30	3,14
	$W = 15$ (S.A.)	1,32	1,23	0,34	2,65
	$W = 10$ (S.A.)	1,62	1,63	0,72	3,75
	$W = 5$ (S.A.)	2,38	2,09	2,89	11,24
Plan 2	$W = 60$ (S.F.)	1,66	1,47	1,57	6,41
	$W = 30$ (S.A.)	0,57	0,50	0,21	2,18
	$W = 25$ (S.A.)	0,60	0,50	0,22	2,21
	$W = 20$ (S.A.)	0,73	0,61	0,31	2,85
	$W = 15$ (S.A.)	1,0	0,84	0,49	3,23
	$W = 10$ (S.A.)	2,72	1,88	10,31	23,83
	$W = 5$ (S.A.)	37,89	36,15	469,62	80,85

Tableau 4.3 – Quelques mesures d'erreur pour la configuration à deux plans avec différentes tailles de fenêtre temporelle W. Les sept lignes du haut correspondant au plan 1, celles du bas au plan 2. S.A. représente le seuillage adaptatif, S.F. l'utilisation d'un seuil fixe.

reprojection des points concordants ayant été assignés au bon plan ne dépasse pas un pixel, alors que celle des points assignés au mauvais plan est d'au plus quatre pixels.

4.3.5 Quadriques

À l'aide de notre environnement virtuel, nous avons projeté la séquence *live band* sur une scène composée de deux sphères. La figure 4.11 présente les résultats pour cinq fenêtres temporelles. La première rangée présente les cartes de disparité, la seconde les correspondances dans l'espace (euclidien) et la troisième l'erreur de reconstruction, en millimètres. Ces résultats montrent que des plus grandes fenêtres temporelles permettent à la fois d'obtenir davantage de (bonnes) correspondances et de réduire l'erreur de reconstruction. Par ailleurs, on remarque que la forme des deux sphères se distingue aisément tant dans la carte de disparité que dans le nuage de points.

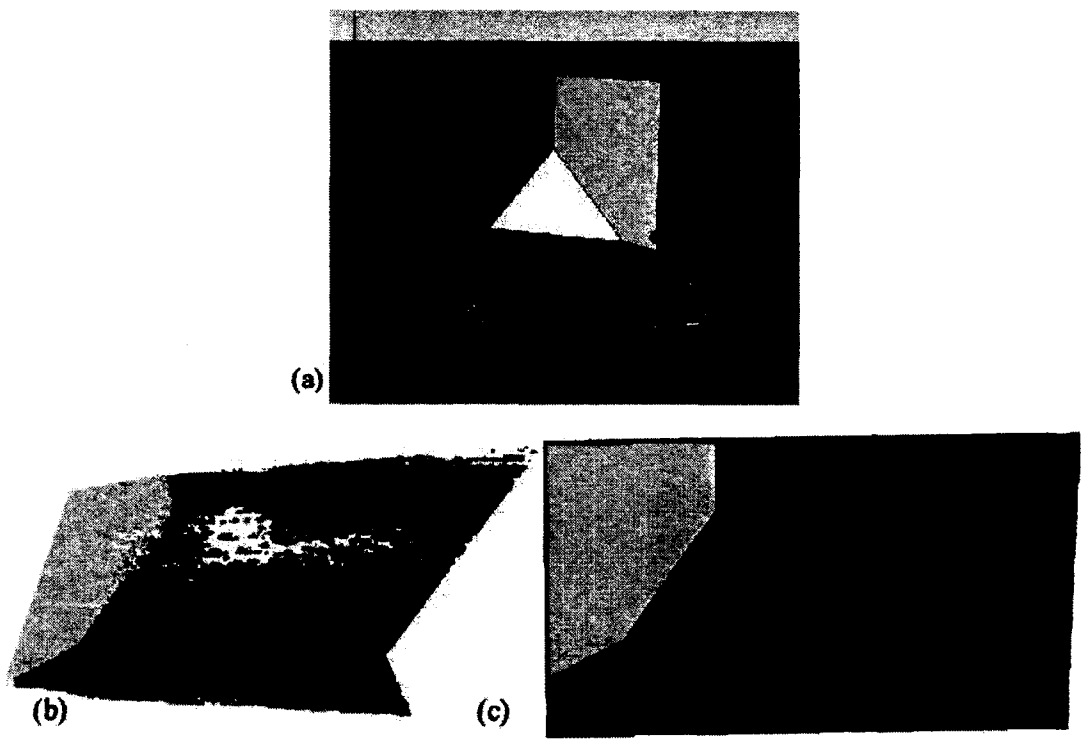


Figure 4.10 – (a) Esquisse de la configuration à quatre plans. (b) Les points 3D retrouvés dans l'espace projectif ($W = 30$) et (c) les quatre plans estimés avec RANSAC.

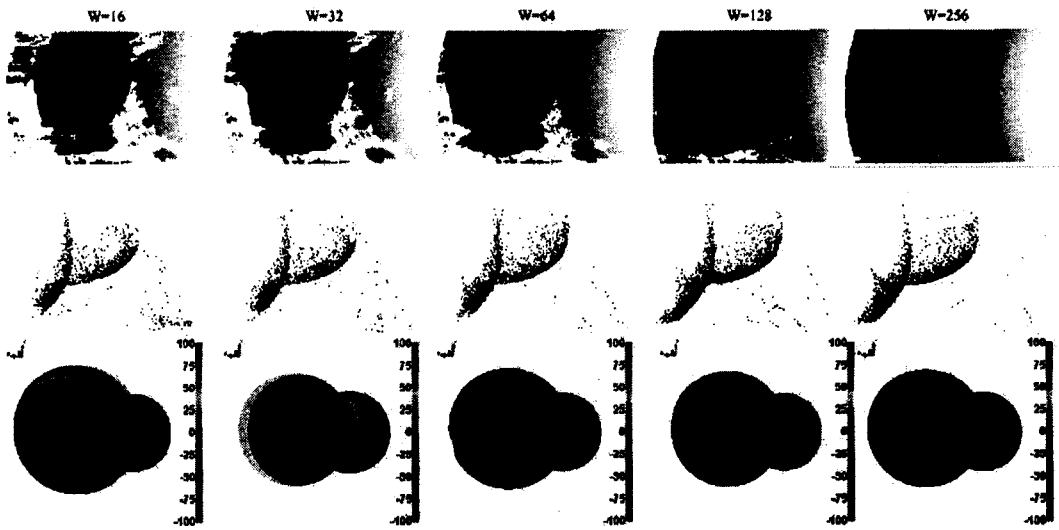


Figure 4.11 – Résultats obtenus en projetant la séquence *live band* sur deux sphères. Des fenêtres temporelles de cinq tailles différentes sont utilisées. La première rangée présente les cartes de disparité obtenues, la seconde montre les points 3D reconstruits, vus en plongée, et la troisième rangée montre l'erreur de reconstruction en millimètres pour les deux sphères. Les sphères ont respectivement des rayons de 1750 et 1105 millimètres et sont situées à 4600 et 4400 millimètres de la caméra.

4.3.6 Espace d'estimation

Tel que mentionné à la section 3.5, les primitives sont estimées dans l'espace projectif, après quoi le résultat est transformé vers l'espace euclidien. La section 3.5 explique que le bruit dans l'espace projectif (x_c, y, x_p) se trouve uniquement sur l'axe x_p. La conversion des points dans l'espace euclidien propage le bruit sur les trois axes. Pour cette raison, l'estimation de primitives est faite dans l'espace projectif. Pour valider ce raisonnement, nous avons utilisé différentes mesures d'erreur quantitatives : la différence angulaire, la différence des paramètres d et le pourcentage de plans bien retrouvés. RANSAC n'étant pas un algorithme déterministe, nous avons pris la moyenne sur 30 exécutions pour l'estimation faite dans les espaces projectif et euclidien, utilisant la distance en Z ou la distance euclidienne classique. Les résultats obtenus (voir tableau 4.4) montrent que l'estimation dans l'espace projectif fonctionne beaucoup mieux. En revanche, le type de distance géométrique utilisé n'a guère d'influence, ce qui confirme qu'on peut ne pas

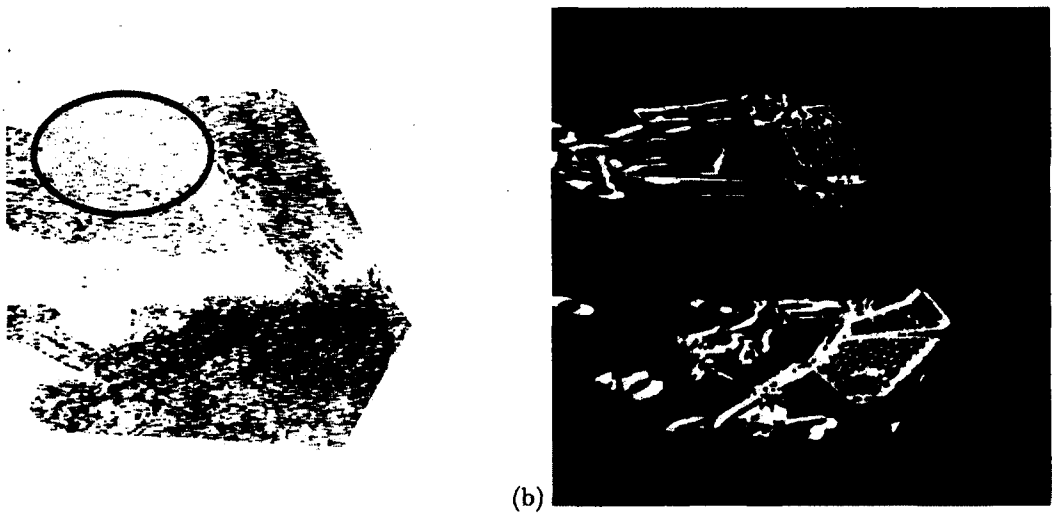
Esp.	$\|\|\cdot\|\|$	Plans	r (%)	$\angle$	$\delta_d(\%)$
Eucl	Z	1	100	0,4993	0,520
Eucl	XYZ	1	100	0,4744	0,497
Proj	Z	1	100	0,1615	0,709
Proj	XYZ	1	100	0,1797	0,710
Eucl	Z	4	25	-	-
Eucl	XYZ	4	25	-	-
Proj	Z	4	90,83	0,5931	1,15
Proj	XYZ	4	82,5	0,6467	1,41

Tableau 4.4 – Pourcentage de succès (r) et mesures d'erreur sur l'angle ($\angle$) et la distance à l'origine (δ_d) pour l'estimation de plans dans les espaces projectif (Proj) et euclidien (eucl). Z et XYZ font référence à la distance le long de l'axe des Z et à la distance euclidienne. Des scènes à un et quatre plans ont été utilisés, avec $W = 45$. Comme on peut le constater, l'estimation dans l'espace euclidien ne permet pas de retrouver tous les plans. Dans l'espace projectif, tous les plans sont bien estimés et l'erreur est plus faible en utilisant la distance le long de l'axe des Z.

considérer les axes x_c et y dans l'espace projectif. Les points 3D utilisés pour réaliser ces tests ont été obtenus en projetant 45 images de la séquence *live band* sur des scènes à un et quatre plans. Un bruit gaussien d'écart-type 8 et une réduction de contraste de 33% ont été appliquées aux images de la caméra.

4.3.7 Sources d'erreur

Le mauvais calibrage du système est une source d'erreur importante. Une mauvais calibrage peut résulter d'un mauvais algorithme, d'une modélisation inadéquate de la caméra ou du projecteur, d'une mauvaise cible de calibrage ou simplement d'une erreur humaine. Le calibrage étant la première étape de notre méthode, une erreur lors de cette phase a des conséquences catastrophiques sur son fonctionnement. Notamment, un mauvais calibrage entraîne une rectification erronée. Puisque l'optimisation par la programmation dynamique établit des correspondances sur les droites épipolaires (horizontales après la rectification), une rectification incorrecte entraîne des mises en correspondance erronées, car le bon correspondant ne se trouve alors plus à la même coordonnée y. En particulier, dans les régions où les pixels actifs sont alignés horizontalement, tous les correspondants potentiels sont à une autre coordonnée y et donc ignorés par



(a) (b)

Figure 4.12 – Image de correspondances (a) et l'image de mouvement qui y est associée (b). En raison d'un mauvais alignement des droites épipolaires et de l'alignement horizontal (fortuit) des pixels actifs, la région encerclée contient très peu de bonnes correspondances.

l'algorithme de programmation dynamique (en vertu de la contrainte épipolaire). La figure 4.12 illustre ce phénomène.

L'erreur de quantification (une image étant composée de pixels, donc discrète) est également à considérer dans l'analyse des performances de notre méthode. Puisque notre méthode établit des correspondances pixel à pixel, elle ne peut être plus précise que l'erreur de quantification, c'est-à-dire un demi-pixel. Nous avons mesuré l'imprécision de la reconstruction 3D réalisée avec la configuration à deux plans pour la comparer à l'erreur de quantification. Le tableau 4.5 présente l'erreur de quantification théorique et l'erreur de reprojection moyenne des plans estimés par notre méthode. Ces résultats montrent que plus W est grand, plus la différence entre l'erreur de reprojection et l'erreur de quantification diminue. Notamment, pour $W = 30$, cette différence est de moins de 0,4 pixels. Ainsi, pour des fenêtres $W \geq 30$, nous affirmons qu'aucune amélioration à la fonction de coût, à la détection de mouvement, à la méthode d'optimisation ou à la méthode de calibration ne pourrait améliorer significativement les résultats. Cette observation signifie qu'il est possible, au moment d'installer notre système, de mesurer l'erreur de quantification pour une séquence vidéo donnée afin d'obtenir une borne inférieure d'erreur.

Fenêtre	Moyenne	Err. quant.
$W = 30$	0,86	0,55
$W = 25$	0,95	0,53
$W = 20$	1,27	0,54
$W = 15$	1,32	0,53
$W = 30$	0,57	0,20
$W = 25$	0,60	0,20
$W = 20$	0,73	0,20
$W = 15$	1,00	0,20

Tableau 4.5 – De gauche à droite : taille de la fenêtre temporelle, l'erreur de reprojection moyene et l'erreur de quantification obtenue lors de la simulation. Les quatre lignes du haut correspondent au plan 1, celles du bas au plan 2. Ces résultats ont été obtenus avec un seuillage adaptatif.

CHAPITRE 5

CONCLUSION ET PERSPECTIVES

Conclusion

Dans ce mémoire a été présentée une méthode de mise en correspondance de pixels basée sur des cooccurrences statistiques d'événements binaires. Cette méthode fonctionne sur un système composé d'une caméra, d'un projecteur et d'une scène complexe. Elle analyse le mouvement dans les images des deux appareils et établit comme correspondants les pixels ayant les profils d'activité les plus similaires. Cette mise en correspondance permet d'obtenir des points 3D. Nous avons également présenté quelques méthodes permettant de décomposer ces points 3D en primitives géométriques (plans, sphères, quadriques) et établi que RANSAC est celle qui fonctionne le mieux.

Une des applications potentielles de notre méthode est de corriger la distorsion induite par la géométrie irrégulière de la scène sur laquelle une séquence vidéo est projetée. Nous avons montré qu'aucune des méthodes de reconstruction 3D ou de mise en correspondance proposées antérieurement ne permet d'arriver efficacement à cette fin. Les principaux avantages de notre méthode sont qu'elle :

- est non intrusive ;

- est robuste aux variations de conditions d'illumination ;
- ne nécessite pas de calibration colorimétrique des appareils ;
- permet un recalibrage du système caméra-projecteur sans devoir interrompre la projection.

Perspectives

Pour l'instant, le développement de la méthode se résume à une preuve de concept. Maintenant que son efficacité est établie, son intégration dans un système réel nécessiterait des travaux supplémentaires. Notamment, la phase de détection de mouvement pourrait être améliorée pour être davantage robuste au bruit. Également, une façon de gérer les scènes mixtes, c'est-à-dire composées de plusieurs types de primitives, devrait être développée. Enfin, il serait nécessaire de savoir déterminer quand la position relative du projecteur et de la scène change afin de pouvoir automatiquement ajuster la transformation de correction de distorsion.

Les performances du système devraient également être améliorées pour permettre son utilisation en temps réel. Pour l'instant, le principal goulot d'étranglement est l'optimisation par programmation dynamique. Des solutions à ce problème existent déjà. D'une part, des implémentations réduisant la complexité algorithmique de la programmation dynamique ont été proposées (voir [15]). D'autre part, l'exécution de cette portion de l'algorithme sur une carte graphique permettrait un gain de temps considérable et se ferait naturellement, puisque la programmation dynamique traite chaque ligne indépendamment. Le processus d'acquisition des images (rectification, détection de mouvement, quantification) est également assez lourd et devrait être accéléré pour permettre un traitement en temps réel. Entre autres, l'ajout du bruit gaussien, dans son implémentation actuelle, est particulièrement lent.

ANNEXE A

Résolution de l'équation de calibrage

En développant l'équation 2.1, on obtient le système suivant :

$$\begin{cases} x &= \frac{p_{11}X+p_{12}Y+p_{13}Z+p_{14}}{p_{31}X+p_{32}Y+p_{33}Z+p_{34}} \\ y &= \frac{p_{21}X+p_{22}Y+p_{23}Z+p_{24}}{p_{31}X+p_{32}Y+p_{33}Z+p_{34}} \end{cases} \quad (A.1)$$

Ce système comporte deux équations et douze variables. Pour le résoudre, il faudrait toutefois obtenir suffisamment d'équations, douze. Toutefois, les inconnues se trouvent tant au numérateur qu'au dénominateur. Il n'est donc pas possible d'exprimer le système sous la forme matricielle $Ax = B$ pour trouver $x = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{34} \end{bmatrix}^T$ en utilisant les méthodes classiques d'algèbre linéaire.

Considérons plutôt la relation entre un pixel de la caméra ou du projecteur p et un point 3D P, en coordonnées homogènes :

$$p_k = \lambda_k J P_k \quad (A.2)$$

dont découle l'équation 2.1. λ est ici le facteur requis pour que la troisième composante de p soit 1. p et JP ne sont donc égaux qu'à un facteur d'échelle près. Ainsi, les vecteurs $\vec{p}$ et $\vec{JP}$

sont parallèles et leur produit vectoriel nul. On a :

$$\begin{aligned} p \times JP &= 0 \\ \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} J_{1.} \cdot P \\ J_{2.} \cdot P \\ J_{3.} \cdot P \end{bmatrix} &= \vec{0} \\ \begin{bmatrix} yJ_{3.} \cdot P - J_{2.} \cdot P \\ J_{1.} \cdot P - xJ_{3.} \cdot P \\ xJ_{2.} \cdot P - yJ_{1.} \cdot P \end{bmatrix} &= \vec{0} \end{aligned} \tag{A.3}$$

L'équation A.3 peut être réécrite sous forme matricielle :

$$\begin{aligned} \begin{bmatrix} \vec{0}^T & -P & yP \\ P & \vec{0}^T & -xP \\ -yP & xP & \vec{0}^T \end{bmatrix} \begin{bmatrix} J_{1.}^T \\ J_{2.}^T \\ J_{3.}^T \end{bmatrix} &= \vec{0}^T \\ Aj &= 0 \end{aligned} \tag{A.4}$$

Dans l'équation A.4, j est un vecteur de 12 éléments et A est une matrice 3×12 de rang 2. En omettant la dernière ligne de A, on a donc 2 équations linéairement indépendantes et 12 variables. Avec 6 paires $\{P, p\}$, on a 12 équations et 12 variables. Ce système est homogène et doit être résolu en faisant une décomposition en valeurs singulières (SVD). Le facteur d'échelle déterminant la matrice de projection s'obtient à l'aide de l'équation A.2.

A.0.8 Résolution de systèmes homogènes

On dit d'un système d'équations linéaires $Ax = b$ qu'il est homogène lorsque le vecteur b est nul. Puisque $A^{-1}\vec{0} = \vec{0}$, on ne peut utiliser la technique habituelle pour trouver x. On doit plutôt faire une décomposition en valeurs singulières (SVD).

La SVD de A est $A = UDV^T$, où U et V sont des matrices orthogonales et D est une matrice diagonale. On note que toute matrice peut être décomposée ainsi. Si les éléments sur la diagonale

de D, appelés les valeurs singulières de A, sont en ordre décroissant, la solution du système est la dernière ligne de V^T. On note que le vecteur x obtenu est unitaire.

ANNEXE B

Détails mathématiques de la rectification d'images

B.1 Matrices de rectification

La rectification est le processus de transformation des images de la caméra C_c et du projecteur C_p pour qu'elle paraissent acquises par une caméra et un projecteur virtuels C_c' et C_p'. Alors que C_c et C_p sont dans des positions quelconques (généralement dans une configuration convergente), C_c' et C_p' sont parallèles. Ainsi, les images rectifiées I_c' et I_p' ont leur droites épipolaires parallèles.

La transformation de rectification correspond à une rotation des appareils sur eux-mêmes. Mathématiquement, on a :

$$\begin{aligned} C_c' &= Q_c C_c \\ C_p' &= Q_p C_p, \end{aligned}$$

où Q_c et Q_p sont des matrices de rotation.

Plusieurs méthodes ont été proposées pour rectifier les images. Certains auteurs utilisent la

matrice fondamentale [36], [38]. L'algorithme d'Isgró et Trucco [29] calcule les matrices de rectification à l'aide de paires de points correspondants, sans que le système ne soit calibré, ni que la matrice fondamentale ne soit connue. L'algorithme présenté ici, proposé par Fusiello et al. [20], calcule les transformations de rectification avec les informations de calibrage des appareils.

Soient K_a, R_a et t_a [1] la matrice de paramètres intrinsèques, la matrice de rotation et le vecteur de translation des appareils non rectifiés. De même, K_a', R_a' et t_a' sont la matrice de paramètres intrinsèques, la matrice de rotation et le vecteur de translation des appareils rectifiés. La figure 2.4 présente un système à deux caméras convergentes et le même système après la rectification des caméras.

De façon absolue, la rectification ne modifie jamais la position des appareils. Toutefois, en raison du changement de repère induit par la rectification, $t_a \neq t_a'$. En fait, puisque t_a et t_a' sont le même point, on a

$$\begin{aligned} {R_p'}^{-1} t_a' &= R_p^{-1} t_a \\ t_a' &= R_p' R_p^{-1} t_a. \end{aligned} \tag{B.1}$$

La section B.1.1 explique comment calculer les matrices R_p' et R_c'. Les matrices K_p' et K_c', quant à elles, peuvent être choisies arbitrairement, tant que

$$\begin{aligned} f_{y_c}' &= f_{y_p}' \\ c_{y_c}' &= c_{y_p}'. \end{aligned}$$

Afin de minimiser la distorsion entre les images originales et les images rectifiées, nous choisissons

$$\begin{aligned} K_p' &= K_p \\ K_c' &= \begin{bmatrix} f_{x_c} & 0 & c_{x_c} \\ 0 & f_{y_p} & c_{x_p} \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

1. Les transformations de rectification de la caméra et du projecteur se calculent exactement de la même façon, à l'aide de leurs informations de calibrage respectives. L'indice a est donc utilisé pour représenter tant la caméra que le projecteur.

Puisque notre système est calibré, K_a, R_a et t_a sont connues pour la caméra et le projecteur. Les matrices de rectification sont alors $Q_a = (K_a{}'R_a{}')(K_aR_a)^{-1}$.

Preuve La projection d'un point P de l'espace sur un pixel s'exprime ainsi :

$$\begin{aligned} p_a &= K_a[R_a|t_a]P \\ &= K_aR_aP + K_at_a \end{aligned}$$

En isolant P, on obtient :

$$P = (K_aR_a)^{-1}(p_a - K_at_a)$$

Sachant que $(AB)^{-1} = B^{-1}A^{-1}$:

$$P = (K_aR_a)^{-1}p_a - R_a^{-1}t_a \tag{B.2}$$

En faisant le même raisonnement dans le repère des appareils rectifiés, on obtient une autre expression pour P :

$$\begin{aligned} p_a{}' &= K_a{}'[R_a{}'|t_a{}']P \\ &= K_a{}'R_a{}'P + K_a{}'t_a{}' \\ P &= (K_a{}'R_a{}')^{-1}(p_a{}' - K_a{}'t_a{}') \end{aligned}$$

Ici, on utilise l'équation B.1 pour obtenir

$$\begin{aligned} P &= (K_a{}'R_a{}')^{-1}p_a{}' - R_a{}'^{-1}R_a{}'R_a^{-1}t_a \\ &= (K_a{}'R_a{}')^{-1}p_a{}' - R_a^{-1}t_a \end{aligned} \tag{B.3}$$

En mettant en relation les équations B.2 et B.3, on obtient :

$$\begin{aligned} (K_aR_a)^{-1}p_a - R_a^{-1}t_a &= (K_a{}'R_a{}')^{-1}p_a{}' - R_a^{-1}t_a \\ (K_aR_a)^{-1}p_a &= (K_a{}'R_a{}')^{-1}p_a{}' \\ p_a{}' &= (K_a{}'R_a{}')(K_aR_a)^{-1}p_a \\ p_a{}' &= Q_ap_a \end{aligned}$$

Ainsi, $Q_a = (K_a{}'R_a{}')(K_aR_a)^{-1}$.

B.1.1 Matrice de rotation

Puisque les axes optiques de la caméra et du projecteur rectifiés sont parallèles, leurs matrices de rotation sont identiques. En faisant coïncider l'origine du système de coordonnées rectifié avec la caméra, on a :

$$\begin{aligned} R_p' &= R_r^{-1} I \\ R_c' &= R_r^{-1} I \end{aligned}$$

où R_r est une matrice de changement de repère. Cette matrice permet de passer du systèmes de coordonnées original au système de coordonnées rectifié. Cette étape est nécessaire, puisque le calcul de Q_p et Q_c se fait dans le système de coordonnées original, alors que les matrices de rotation des appareils rectifiés sont connues dans le système rectifié. R_r est une matrice 3×3 dont les colonnes sont les vecteurs r_1, r_2, r_3 formant la base du système rectifié, exprimés dans le système original :

$$R_r = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} \\ r_{1y} & r_{2y} & r_{3y} \\ r_{1z} & r_{2z} & r_{3z} \end{bmatrix}.$$

Dans le repère rectifié, la position de la caméra et du projecteur ne diffèrent que par une translation en x (de par la définition de la rectification). Dans le repère original, on a donc :

$$r_1 = \frac{C_c - C_p}{||C_c - C_p||}.$$

r_1 étant connu, on complète la matrice à l'aide de produits vectoriels :

$$\begin{aligned} r_2 &= \frac{r_1 \times r_k}{||r_1 \times r_k||} \\ r_3 &= r_1 \times r_2, \end{aligned}$$

où r_k est un vecteur arbitraire non parallèle à r_1. Afin de s'assurer que la caméra rectifiée voie la scène, nous choisissons r_k comme étant la composante z de R_c. Également, r_3 n'a pas à être normalisé, puisque le produit vectoriel de deux vecteurs unitaires est toujours unitaire.
On note qu'en plus d'intervenir dans le calcul des matrices de rectification, R_r et R_r^{-1} permettent de transformer des points et primitives du système original vers le système rectifié et vice-versa.

B.2 Autres considérations sur la rectification

Paramètres intrinsèques Les paramètres intrinsèques des appareils rectifiés ne sont pas les mêmes que ceux des appareils originaux. Pour assurer le parallélisme des droites épipolaires, on doit avoir

$$\begin{aligned} {f_{y_c}}' &= {f_{y_p}}' \\ {c_{y_c}}' &= {c_{y_p}}' \end{aligned}$$

Les deux autres paramètres intrinsèques peuvent être choisis arbitrairement. Par soucis de fidélité aux images originales, les mêmes f_{x_a} et c_{x_a} qu'ont les appareils originaux sont attribués aux appareils rectifiés.

Dimensions Les dimensions des images rectifiées sont aussi différentes de celles des images originales. En particulier, la résolution en y doit être la même dans ${C_c}'$ et ${C_p}'$. Le choix judicieux des matrices ${K_a}'$ garantit que cette condition soit remplie. Les dimensions des images rectifiées peuvent être calculées en appliquant la transformation de rectification aux coins des images originales et en prenant le plus petit rectangle contenant les quatre points ainsi obtenus. Un décalage doit également être effectué pour ramener l'origine des images rectifiées au pixel $(0, 0)$. De plus, puisque la rectification est une rotation quelconque autour d'un axe quelconque, la position des coins n'est pas nécessairement préservée, comme le montre la figure B.1.

Implémentation Lors de l'implémentation d'un algorithme de rectification, la transformation de chaque pixel non rectifié avec ${p_a}' = Q_a p_a$ ne garantit pas qu'une valeur soit affectée à chaque pixel de l'image rectifiée. Il est donc préférable d'utiliser la rectification inverse Q_a^{-1} pour obtenir l'intensité de chaque pixel de l'image rectifiée avec $p_a = Q_a^{-1} {p_a}'$. Si les coordonnées de p_a ne sont pas entières, on interpole bilinéairement pour calculer l'intensité. Si ces coordonnées sont à l'extérieur de l'image originale, on affecte au pixel rectifié une intensité de remplissage choisie arbitrairement (voir figure B.1).

Figure B.1 – La recherche des dimensions de l'image rectifiée. On remarque que la position des coins est modifiée par la rectification. Une couleur arbitraire est utilisée pour remplir la portion inutilisée de l'image rectifiée.

ANNEXE C

Les espaces projectif et euclidien

C.1 Conversion de points de l'espace projectif vers l'espace euclidien

On a un système constitué d'un projecteur et d'une caméra. Définissons la position de la caméra comme étant l'origine du système. La relation entre les pixels p_c de l'image de la caméra et les points $P = (X, Y, Z)$ de l'espace est donc :

$$p_c = [A_c]\,[I|0]\,P/Z$$

De même, la relation entre les pixels projetés p_p et les (mêmes) points P de l'espace est :

$$p_p = [A_p]\,[R|T]\,P/Z$$

Explicitement, ces équations s'écrivent :

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} f_{x_c} & 0 & c_{x_c} \\ 0 & f_{y_c} & c_{y_c} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} /Z \tag{C.1}$$

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \lambda_p \begin{bmatrix} f_{x_p} & 0 & c_{x_p} & 0 \\ 0 & f_{y_p} & c_{y_p} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r1 & r2 & r3 & t_x \\ r4 & r5 & r6 & t_y \\ r7 & r8 & r9 & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Notre système traite des images rectifiées, donc uniquement une translation en x sépare le projecteur de la caméra. La dernière équation se simplifie ainsi à :

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_{x_p} & 0 & c_{x_p} \\ 0 & f_{y_p} & c_{y_p} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} /Z \tag{C.2}$$

En développant l'équation C.1, on obtient les équations suivantes :

$$\begin{aligned} x_c &= (X f_{x_c} + Z c_{x_c})/Z \\ y_c &= (Y f_y + Z c_y)/Z \end{aligned}$$

que l'on peut transformer en

$$X = \frac{Z \cdot (x_c - c_{x_c})}{f_{x_c}} \tag{C.3}$$

$$Y = \frac{Z \cdot (y_c - c_y)}{fy} \tag{C.4}$$

De même, l'équation C.2 se développe en

$$x_p = (X f_{x_p} + Z c_{x_p} + t_x f_{x_p})/Z \tag{C.5}$$

En remplaçant dans C.5 X par son expression (C.3), on obtient

$$x_p = \left[\frac{Z(x_c - c_{x_c})}{f_{x_c}} \cdot f_{x_p} + Z c_{x_p} + t_x f_{x_p} \right] / Z \tag{C.6}$$

où la seule inconnue est Z. On peut alors obtenir une expression directe pour Z à l'aide de quelques manipulations algébriques élémentaires.

$$Z = \frac{t_x f_{x_p} f_{x_c}}{f_{x_c}(x_p - c_{x_p}) - f_{x_p}(x_c - c_{x_c})} \tag{C.7}$$

Il ne reste qu'à remplacer Z dans C.3 et C.4 par son expression en C.7 pour obtenir une expression pour X et Y et ainsi compléter le système.

$$X = \frac{t_x f_{x_p}}{f_{x_c}(x_p - c_{x_p}) - f_{x_p}(x_c - c_{x_c})} \cdot (x_c - c_{x_c}) \tag{C.8}$$

$$Y = \frac{t_x f_{x_p} f_{x_c}}{f_{x_c}(x_p - c_{x_p}) - f_{x_p}(x_c - c_{x_c})} \cdot (y_c - c_{y_c}) \cdot \frac{1}{f_y} \tag{C.9}$$

$$Z = \frac{t_x f_{x_p} f_{x_c}}{f_{x_c}(x_p - c_{x_p}) - f_{x_p}(x_c - c_{x_c})}$$

C.2 Matrice de conversion

Il est possible d'effectuer le passage de l'espace projectif à l'espace euclidien avec une multiplication matricielle. En effet, en multipliant les équations C.8 et C.7 par $\frac{f_y}{f_y}$, elles ont un dénominateur commun avec C.9. La division perspective par la composante w tient alors le rôle de ce dénominateur. On obtient ainsi la matrice de conversion suivante :

$$T = \begin{bmatrix} t_x f_{x_p} f_y & 0 & 0 & -t_x f_{x_p} f_y c_{x_c} \\ 0 & t_x f_{x_p} f_{x_c} & 0 & -t_x f_{x_p} f_{x_c} c_{y_c} \\ 0 & 0 & 0 & t_x f_{x_p} f_{x_c} f_y \\ -f_{x_p} f_y & 0 & f_{x_c} f_y & f_y(f_{x_p} c_{x_c} - f_{x_c} c_{x_p}) \end{bmatrix}$$

et les points de l'espace projectif peuvent être transformés vers l'espace euclidien de la façon suivante :

$$\begin{bmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{bmatrix} = T * \begin{bmatrix} x_c \\ y \\ x_p \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{bmatrix} /W_h$$

C.3 Conversion de plans de l'espace projectif vers l'espace euclidien

Comme les points, il est possible de convertir l'équation d'un plan π_p de l'espace projectif à l'espace euclidien pour y donner le plan π_E. Étant donnés des points P_E et P_p situés respectivement sur π_E et π_p et une matrice T de format 4×4 telle que $P_E = TP_p$, alors [1]

$$\pi_E = T^{-t}\pi_p.$$

Preuve Soient P_{1E}, P_{2E}, P_{3E} trois points non colinéaires de l'espace euclidien, définissant un plan π_E. Soit également une matrice T décrivant une transformation projective telle que $P_{iE} = TP_{ip}$, où $i \in [1, 3]$ et les P_{ip} sont des points de l'espace projectif. Puisque T préserve la colinéarité, les points P_{ip} définissent un plan π_p dans l'espace projectif. En groupant les points

1. L'exposant $-T$ indique la transposée de l'inverse de la matrice à laquelle il est affecté.

P_{iE} et P_{ip} dans des matrices M_E et M_p (en colonnes), on a donc :

$$\pi_E^t M_E = 0 \tag{C.10}$$

$$\pi_p^t M_p = 0 \tag{C.11}$$

$$M_E = T M_p \tag{C.12}$$

De l'équation C.12, on peut déduire que

$$M_p = T^{-1} M_E$$

On remplace ensuite M_p dans C.11 pour obtenir

$$\begin{aligned} \pi_p^t T^{-1} M_E &= \vec{0} \\ (\pi_p^t T^{-1}) M_E &= \vec{0} \end{aligned} \tag{C.13}$$

Puisque l'inverse d'une transformation projective est aussi une transformation projective, $\pi_p^t T^{-1}$ définit un plan. P_{1E}, P_{2E} et P_{3E} n'étant pas colinéaires, ils définissent un et un seul plan. De C.10 et de ce résultat, on conclut que

$$\begin{aligned} \pi_E^t &= \pi_p^t T^{-1} \\ \pi_E &= T^{-t} \pi_p \end{aligned}$$

On note enfin que l'hypothèse de non colinéarité des points P_{iE} n'est pas restrictive, car tout plan non dégénéré contient une infinité de points non tous colinéaires.

C.4 Conversion de quadriques de l'espace projectif vers l'espace euclidien

De même, les quadriques peuvent aisément être converties d'un espace à l'autre avec la formule suivante :

$$Q_E = T^{-T} Q_p T^{-1}.$$

Preuve Une quadrique Q est définie comme étant la surface telle que $P^T Q P = 0$. On a alors

$$P_p^T Q_p P_p = 0 \tag{C.14}$$

$$P_E^T Q_E P_E = 0 \tag{C.15}$$

En introduisant la relation $P_E = T P_p$ dans l'équation C.15, la conclusion s'ensuit naturellement :

$$\begin{aligned} (TP_p)^T Q_E T P_p &= 0 \\ P_p^T (T^T Q_E T) P_p &= 0 \\ T^T Q_E T &= Q_p \\ Q_E &= T^{-T} Q_p T^{-1} \end{aligned}$$

Remarque On note que le raisonnement précédent est valable pour toute transformation Ξ appliquée sur une quadrique. Plus généralement, on obtient

$$Q' = \Xi^{-T} Q \Xi^{-1}$$

ANNEXE D

Chur [1] : Environnement de projection virtuel

Notre méthode se base sur un système où un projecteur, une caméra et une scène complexe sont présents. De plus, la caméra et le projecteur doivent être parfaitement synchronisés. En raison de ces contraintes, nous avons choisi de bâtir un environnement de projection virtuel, rendant ainsi possibles des démonstrations impromptues de l'application de notre méthode. Les tests en sont par le fait même grandement facilités, car les données de calibrage sont connues exactement. Un source d'erreur potentielle est ainsi supprimée. De plus, différentes scènes peuvent être générées et les cartes de disparité étalon correspondantes obtenues facilement.

D.1 Contexte technique

L'environnement a été programmé en C++ avec la librairie Qt. Le rendu est fait avec OpenGL, via l'interface Qt. L'interface graphique a été conçue avec l'outil *Qt Designer*.

1. *Chur*, ville de Suisse et chef-lieu du canton de Graubünden. Le choix de ce nom est une coquetterie de la part de l'auteur.

D.2 Matériel et logiciels requis

Compilation

Les éléments suivants doivent être installés pour pouvoir compiler Chur :

- Qt, version 4.6.1 ou plus récente ;
- OpenCV, version 2.0 ou plus récente ;
- un compilateur C++ supportant OpenMP ;
- les librairies TNT et JAMA, version 1.2.6 ou plus récente ;
- la librairie CMinPack, version 1.0.2 ou plus récente.

Exécution

Chur peut fonctionner sur tout PC équipé de Windows Vista ou d'une version plus récente. Il fonctionnera également sous Windows XP si les libraires redistribuables de C++ y sont installées. Une carte graphique supportant les nuanceurs programmables est également requise. Il est recommandé de disposer d'au moins 1 Go de mémoire vive. Chur fonctionnera avec moins, mais la durée des projections pouvant être réalisées sera limitée. Certaines fonctionnalités dépendent de la librairie OpenCV. Dans une version ultérieure, cette dépendance pourrait être éliminée, sauf pour l'utilisation de la caméra. Les fichiers binaires (*.dll*) requis sont fournis avec l'application.

D.3 Portabilité

À ce jour, Chur n'a pas été porté sur d'autres plateformes que Windows. Il est permis de croire qu'il serait plutôt simple de le faire, du moins pour les plateformes supportées par Qt et OpenCV (parmi lesquelles, notamment, Linux/Unix et MacOS).

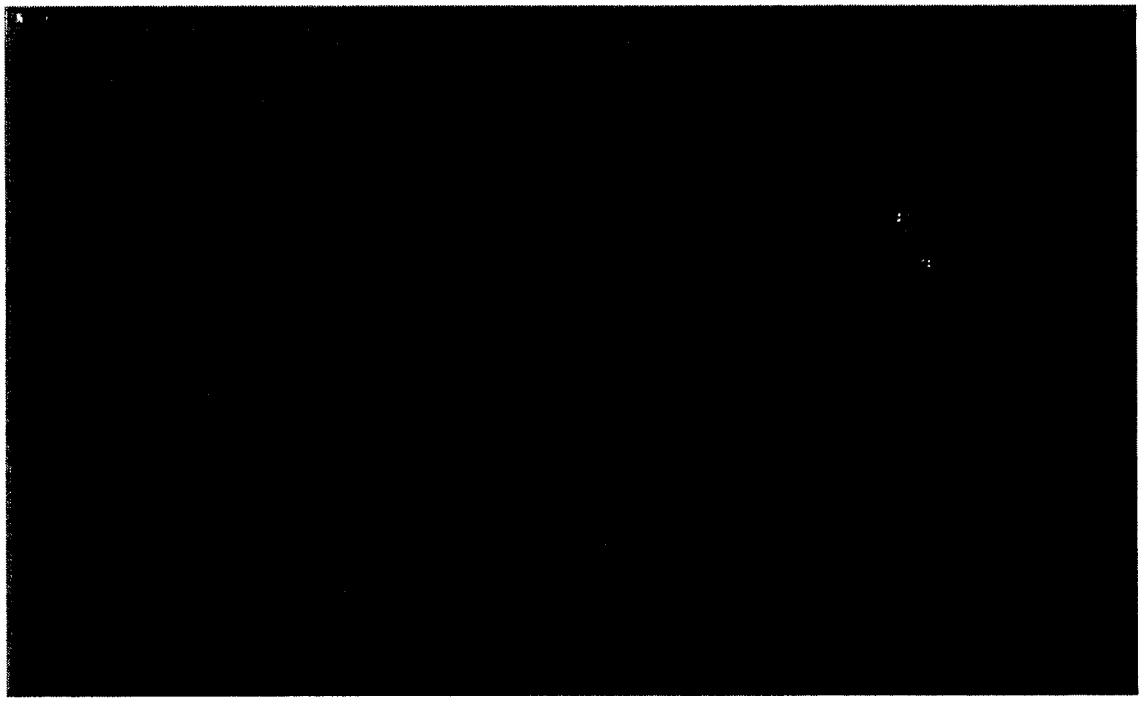


Figure D.1 – L'interface principale de notre environnement de projection virtuel.

D.4 Guide d'utilisation

Onglet *Configuration*

L'interface graphique principale (voir figure D.1) est divisée en trois parties : la configuration du test à réaliser, la configuration des appareils et la visualisation.

Configuration des tests

Le cadre dans le coin supérieur gauche regroupe les champs et boutons permettant de configurer les tests à exécuter.

Configurations préenregistrées Une configuration peut être enregistrée dans un fichier xml dont l'extension est *.chur*. Le format de ce fichier est défini à la section D.5. Lors du lancement, Chur tente de charger le fichier passé en paramètre au programme, ou *defaults.chur* si aucun paramètre n'est spécifié. Le bouton *Valeurs par défaut* permet de charger une configuration lors de l'exécution.

Scènes Les scènes utilisées sont chargées depuis un fichier xml d'extension *.scn*. Le format de ce fichier est défini à la section D.6. Dans la version actuelle de Chur, il n'existe pas d'outil interactif d'édition de scène. Les scènes doivent donc être créées à la main, souvent au prix de nombreux essais et erreurs.

Projection/Estimation Un bouton radio permet de choisir si une projection (*Caméra*, *Fichier*) ou l'estimation de primitives (*Points*) doit être faite. Dans le premier cas, les images proviennent soit d'une caméra branchée à l'ordinateur, si disponible (OpenCV est requis pour l'utilisation de la caméra) ou de fichiers. Les images de la vidéo doivent alors déjà être dans des fichiers séparés, numérotés séquentiellement. Les formats de fichier supportés sont ceux supportés par Qt (tous les formats courants).
Dans le cas de l'estimation de primitives, le fichier a l'extension *.pts* et contient une liste de points (x_c, y, x_p) (un point par ligne, les composantes sont séparées par des espaces). De tels fichiers sont produits par Chur à la suite de la mise en correspondance (voir paragraphe *Carte de disparité*).

Durée de la projection Si cette option est sélectionnée, la projection cessera après que le nombre spécifié d'images aura été atteint.

Nombre d'images requises pour la mise en correspondance Définition de la fenêtre temporelle W.

Pas à pas Si cette option n'est pas sélectionnée, la mise en correspondance commence automatiquement lorsque W images sont disponibles, après quoi l'estimation de primitives est également faite automatiquement. Lorsque l'option *Pas à pas* est sélectionnée, l'utilisateur doit déclencher manuellement chacune de ces étapes.

Carte de disparité Si cette option est sélectionnée, une carte de disparité et un fichier *.pts* contenant les points (x_c, y, x_p) résultant de la mise en correspondance seront produits au terme de celle-ci. Par défaut, ces fichiers sont enregistrés sous les noms *dispmap.ppm* et *dispmap.pts*.

Échantillonnage Il est possible de ne pas utiliser tous les points résultant de la mise en correspondance lors de l'estimation de primitives, afin d'accélérer le traitement. Pour autant de points que spécifié ici (ν), un seul sera utilisé. (Par exemple, si $\nu = 1$, tous les points sont utilisés. Si $\nu = 4$, 25 % des points sont utilisés (1 point sur 4).)

Nombre et type de primitives Il est nécessaire de spécifier le nombre et le type de primitives présents dans la scène. Dans la version courante de Chur, les scènes mixtes ne sont pas supportées.

Correction de la distorsion Après que l'estimation de primitives a été faite, il est possible de prédéformer les images à projeter afin d'éliminer la distorsion induite par la scène. La prédéformation peut être faite en fonction de deux points de vue : celui de la caméra qui filme la scène (et dont la position par rapport au projecteur est fixe) ou celui de la caméra externe, simulant la position d'un observateur.

Supprimer les images, les correspondances Ces boutons permettent de vider la cache d'images et de correspondances. Il est à noter que ces caches sont vidées automatiquement lorsque requis, lors du changement de séquence ou d'une nouvelle mise en correspondance, par exemple. Les points chargés depuis des fichiers *.pts* ne sont pas conservés dans cette cache.

Projection, Correspondances, Estimation, Interruption Ces boutons permettent de lancer ou d'interrompre une des phases de traitement de notre méthode. Chaque bouton n'est activé que lorsqu'il est possible d'accomplir l'action qui y est reliée.

Configuration des appareils

Au centre de la fenêtre se trouve la zone de configuration des appareils.

Distance caméra-projecteur Distance, en millimètres, entre la caméra et le projecteur. Par défaut, le projecteur est à droite de la caméra. Une valeur négative le placera plutôt à gauche de la caméra.

Angle de la caméra, du projecteur Par défaut, les deux appareils regardent dans la direction de l'axe des Z négatif. Un angle positif entraîne une rotation de l'appareil autour de l'axe des Y dans le sens antihoraire, un angle négatif dans le sens horaire. Uniquement un balayage horizontal est possible, tant pour la caméra que pour le projecteur.

Paramètres intrinsèques Le modèle utilisé pour la caméra et le projecteur est le même. Les matrices de paramètres intrinsèques ont la forme suivante :

$$\begin{bmatrix} f_x & [s=0] & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

où les paramètres sont les suivants :

- (s_x, s_y) : les dimensions d'un pixel (en millimètres)
- f : la distance focale (en millimètres)
- (f_x, f_y) : $(\frac{f}{s_x}, \frac{f}{s_y})$
- (c_x, c_y) : le centre de projection (en pixels)

– s : le facteur de cisaillement (doit être nul, dans la version courante)

Dégradation de contraste La valeur spécifiée est le niveau de contraste dans l'image de la caméra. 1 correspond à l'image originale, 0 à une image complètement noire. Il est à noter que l'effet de la dégradation de contraste n'est visible que dans le mode caméra (voir D.4).

Bruit gaussien La valeur spécifiée est l'écart-type du bruit gaussien (de moyenne 0) ajouté à l'image de la caméra. Le bruit est ajouté par pixel. Il est à noter que l'effet du bruit gaussien n'est visible que dans le mode caméra (voir D.4).

Visualisation

La portion de droite de l'interface graphique permet de voir la scène, les points et les primitives reconstruits. Cet affichage est interactif et l'utilisateur peut le configurer en utilisant le clavier, la souris et les boutons et cases à cocher présentes sur l'interface graphique.
Les primitives et points reconstruits sont de la même couleur que les objets originaux auxquels ils sont respectivement associés. De plus, des mesures quantitatives d'erreur d'estimation sont affichées dans le coin inférieur gauche de l'interface graphique.

Modes *Scène*, *Caméra* et *Projecteur* Le mode définit le point de vue de l'utilisateur. Les modes *Caméra* et *Projecteur* positionnent l'utilisateur exactement à la même position que la caméra et le projecteur (respectivement). Le mode *Scène* donne à l'utilisateur un point de vue externe de la scène et des appareils. En mode *Scène*, le point de vue peut être modifié avec la souris (voir paragraphe *Souris*).

Souris La souris permet de modifier le point de vue de l'utilisateur (« l'oeil ») lorsque le mode *Scène* est sélectionné. L'oeil est positionné sur une sphère dont le centre est par défaut la caméra et est toujours orienté vers le centre de cette sphère. Le bouton de gauche contrôle

la position de l'oeil sur la sphère, c'est-à-dire sa rotation autour de la caméra. Le bouton de droite contrôle la translation du centre de la sphère. La roulette contrôle le rayon de la sphère, c'est-à-dire le facteur de grossissement.
Le bouton du milieu permet d'appliquer aux objets de la scène une rotation autour de la caméra. Cette rotation modifie effectivement la scène et fonctionne dans les trois modes. Déplacer les objets lorsqu'un traitement est en cours risque de fausser les calculs d'estimation ou de mise en correspondance. Dans la version courante de Chur, il n'est pas possible d'éloigner ou de rapprocher les objets de la caméra, de les translater, ni de sauvegarder leur nouvelle position. Pour ces raisons, il est recommandé d'utiliser cette fonctionnalité avec précaution.

Clavier Les touches F1-F12 permettent de choisir lesquels des nuages de points reconstruits (segmentés par primitives) doivent être visibles. Ces touches n'ont un effet que lorsque l'option *Afficher les points reconstruits* est sélectionnée.
Les combinaisons MAJ + F1-F12 permettent de choisir lesquelles des primitives reconstruires doivent être visibles. Ces combinaisons de touches n'ont un effet que lorsque l'option *Afficher les primitives reconstruites* est sélectionnée.

Point de vue initial Le bouton *Point de vue initial* permet de restituer l'oeil à sa position par défaut.

Afficher les primitives reconstruites Cette option contrôle l'affichage des primitives reconstruites. Son effet n'est visible qu'après que l'estimation des primitives est terminée. Les combinaisons MAJ + F1-F12 permettent d'afficher les primitives individuellement.

Activer le projecteur Cette option permet « d'allumer » ou « d'éteindre » le projecteur virtuel. Si le projecteur virtuel est « éteint », la caméra n'accumulera pas de données et il sera impossible d'effectuer la mise en correspondance.

Cacher les objets Cette option permet de désactiver l'affichage des objets de la scène, c'est-à-dire des primitives à reconstruire.

Cacher la caméra et le projecteur Cette option permet de désactiver l'affichage physique de la caméra et du projecteur (mode *Scène* seulement). Désactiver l'affichage de la caméra et du projecteur n'altère en rien leur fonctionnement.

Afficher les points reconstruits Cette option contrôle l'affichage des points reconstruits. Si seulement la mise en correspondance a été faite, les points sont tous de la même couleur. Si l'estimation des primitives a été faite, chaque point est de la couleur de la primitive originale à laquelle il est associé. Un nuage de points supplémentaire (d'une couleur arbitraire, non utilisée par les objets de la scène) contient les points aberrants. Les touches F1-F12 permettent d'afficher les nuages de points individuellement.
Il est à noter que la valeur d'échantillonnage spécifiée (voir D.4) a un impact sur la quantité de points affichés.

Taille des points La valeur spécifiée ici contrôle la taille des points.

Onglet *Détails*

L'onglet *Détails* permet de voir les images de la caméra et du projecteur aux différentes étapes de traitement, entre la projection et leur analyse lors de la mise en correspondance. Les étapes disponibles sont, pour chaque appareil :

- l'image originale ;
- l'image rectifiée ;
- l'image à niveau de gris (étape préalable à la quantification) ;
- l'image quantifiée ;
- l'image de mouvement.

D.5 Format des fichiers de configuration (*.chur*)

Les noms de fichiers doivent contenir le chemin d'accès, au besoin. Le chemin d'accès peut être relatif au dossier courant. Les angles doivent être spécifiés en degrés.

<chur>

<defaults>

<scene></scene>

Nom du fichier contenant la scène à charger.

<sequence_type></sequence_type>

0 : Projeter des images à partir de fichiers

1 : Projeter les images d'une caméra connectée à l'ordinateur

2 : Estimer les primitives d'après les points contenus dans le fichier .pts spécifié

<sequence_filename></sequence_filename>

Nom du premier fichier image à projeter ou du fichier .pts à utiliser pour l'estimation (selon la valeur de *sequence-type*)

<view_type></view_type>

0 : Mode Caméra

1 : Mode Projecteur

2 : Mode Scène

<number_frames></number_frames>

Taille de la fenêtre W. Valeurs entières positives seulement.

<quantization_levels></quantization_levels>

Nombre de niveaux de quantification. Valeurs entières positives seulement.

<nb_primitives></nb_primitives>

Nombre de primitives à estimer. Valeurs entières positives seulement.

`<sampling></sampling>`
Valeur d'échantillonnage. Valeurs entières positives seulement.

`<primitive_type></primitive_type>`
12 : Les primitives à estimer sont des plans

9 : Les primitives à estimer sont des sphères

`<save_disparity_map></save_disparity_map>`
1 : Sauvegarder la carte de disparité et le fichier .pts au terme de la mise en correspondance

0 : Ne pas sauvegarder la carte de disparité et le fichier .pts

`<show_points></show_points>`
1 : Afficher les points reconstruits

0 : Ne pas afficher les points reconstruits

`<stepbystep></stepbystep>`
1 : Mode Pas à pas activé

0 : Mode Pas à pas désactivé

`<reconstruction_visibility></reconstruction_visibility>`
1 : Afficher les primitives reconstruites

0 : Ne pas afficher les primitives reconstruites

`<projector_enabled></projector_enabled>`
1 : « Allumer » le projecteur

0 : « Éteindre » le projecteur

`<hide_camproj></hide_camproj>`
1 : Cacher la caméra et le projecteur

0 : Afficher la caméra et le projecteur

`<hide_objects></hide_objects>`

1 : Cacher les objets de la scène

0 : Afficher les objets de la scène

`<max_number_frames></max_number_frames>`

Nombre maximal d'images à projeter (-1 : aucune limite).

`<length_limit_status></length_limit_status>`

1 : Limiter le nombre d'images à projeter

0 : Ne pas limiter le nombre d'images à projeter

`<camera fx="579.411" fy="579.411" cx="320" cy="240" />`

Paramètres intrinsèques de la caméra. (Les valeurs indiquées sont les valeurs par défaut.)

`<projector fx="500" fy="375" cx="160" cy="120" />`

Paramètres intrinsèques du projecteur. (Les valeurs indiquées sont les valeurs par défaut.)

`<tx></tx>`

Distance caméra-projecteur

`<projector_angle></projector_angle>`

Angle du projecteur

`<camera_angle></camera_angle>`

Angle de la caméra

`<camera_field_of_view_y></camera_field_of_view_y>`

Angle correspondant à l'ouverture verticale de la caméra (Défaut : 45)

`<point_size></point_size>`

Taille des points (défaut : 1). Valeurs entières positives seulement.

`<contrast_degradation_factor></contrast_degradation_factor>`

Contraste de l'image de la caméra. Valeurs entre 0 et 1.

`<gaussian_noise_stddev></gaussian_noise_stddev>`

Écart-type du bruit gaussien ajouté à l'image de la caméra (0 : aucun bruit)

`<distortioncorrection></distortioncorrection>`

1 : Utiliser la prédéformation des images

0 : Ne pas l'utiliser

`<distortion_correction_type></distortion_correction_type>`

0 : Prédéformer les images pour le point de vue de la caméra

2 : Prédéformer les images pour le point de vue de l'utilisateur (mode *Scène*)

`<configuration_name></configuration_name>`

Nom donné à la carte de disparité et au fichier .pts sauvegardés. (Défaut : dispmap)

`</defaults>`

`</chur>`

D.6 Format des fichiers de scène (*.scn*)

L'unité utilisée dans les fichiers de scène est le mètre. Les translations sont effectuées avant les rotations. Les rotations autour des trois axes sont effectuées dans l'ordre x, y, puis z. Une scène peut contenir autant de primitives que désiré. Il est à noter que, bien que tous les types de primitives soient ici dans une même scène, la version courante de Chur ne permet pas de reconstruire des primtives différentes dans une même scène. Il est néanmoins possible de charger et d'afficher de telles scènes.

`<scene name="defaultscene">`

Par défaut, un plan est un quadrilatère, défini ici par ses quatre coins. *tl*, *tr*, *bl*, *br* correspondent respectivement aux coins supérieurs gauche et droit et inférieurs gauche et droit.

```
<plane
 tlx="-5" tly=" 5" tlz="-3.5"
 trx=" 5" try=" 5" trz="-3.5"
 blx="-5" bly="-5" blz="-3.5"
 brx=" 5" bry="-5" brz="-3.5"
 rx="-25"
 ry="-25"
 rz="0"
 tx="0"
 ty="0"
 tz="0" >
 <colour red="0" green="0" blue="255" />
</plane>
```

Il est également possible de définir des plans triangulaires.

```
<triangle
 p1x="0.5" p1y="2.5" p1z="-2.5"
 p2x="2.5" p2y="1."  p2z="-2.5"
 p3x="2.5" p3y="2.5" p3z="-1."
 tx="-0.2"
 ty="1.4"
 tz="0."
 rx="10"
 ry="32"
 rz="-90" >
```

```
  <colour red="0" green="128" blue="255" />
 </triangle>
```

Les sphères sont définies par la position du centre et leur rayon.

```
 <sphere
  centerx="-0.05"
  centery="0."
  centerz="-4.6"
  radius ="2.5" >
  <colour red="200" green="100" blue="100" />
 </sphere>
```

Il est également possible de définir des cylindres, même si Chur ne permet pas pour l'instant d'en faire l'estimation. *theta* et *phi* sont les angles de rotation et d'élévation de l'axe principal du cylindre.

```
 <cylinder
  centerx="0.05"
  centery="-2.5"
  centerz="-3.10"
  radius ="2.25"
  height ="9.5"
  theta  ="60"
  phi    ="45">
  <colour red="200" green="100" blue="100" />
 </cylinder>
</scene>
```

D.7 Bogues

Il est possible que Chur contienne quelques bogues. Voici la liste des bogues connus :

- Il semble arriver que des boutons soient désactivés alors qu'ils ne devraient pas l'être.
- La modification des paramètres f_y ou c_y de la caméra ou du paramètre c_y du projecteur entraînent une mauvaise rectification et conséquemment l'échec des étapes subséquentes de l'algorithme (voir section 4.3.7).
- Lorsque la correction de la distorsion est activée, l'alignement horizontal des portions d'images n'est pas toujours correct.

Bibliographie

[1] M.-F. AUCLAIR-FORTIER : Extraction de caractéristiques : Contours multispectraux, contours de texture et routes. Mémoire de maîtrise, Université de Sherbrooke, Sherbrooke, Canada, 1999.

[2] Mr. BEAM : http ://www.mr-beam.nl/, 2011.

[3] R. BELLMAN : The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–515, 1954.

[4] Y. BENEZETH, P.-M. JODOIN, V. SALIGRAMA et C. ROSENBERGER : Abnormal events detection based on spatio-temporal co-occurences. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2458–2465, 2009.

[5] L. BODUM, J. OVERBY, E. KJEMS et P. M. ILSØE : Automatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. Dans *Proceedings of the* 20^{th} *International Congress for Photogrammetry and Remote Sensing*, pages 1–6, 2004.

[6] Yuri BOYKOV et Vladimir KOLMOGOROV : An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1124–1137, 2004.

[7] Yuri BOYKOV, Olga VEKSLER et Ramin ZABIH : Fast approximate energy minimization

via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239, 2001.

[8] C. BRENNER : Towards fully automatic generation of city models. Dans *Proceedings of International Archives of Photogrammetry and Remote Sensing*, pages 85–92, 2000.

[9] D. COTTING, M. NAEF, M. GROSS et H. FUCHS : Embedding imperceptible patterns into projected images for simultaneous acquisition and display. Dans *Proceedings of the* 3^{rd} *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 100–109, 2004.

[10] F. DELLAERT, S. M. SEITZ, C. E. THORPE et S. THRUN : Structure from motion without correspondence. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pages 557–564, 2000.

[11] M.-A. DROUIN, P.-M. JODOIN et J. PRÉMONT : Camera-projector matching using an unstructured video stream. Dans *Proceedings of the IEEE International Workshop on Projector-Camera Systems*, 2010.

[12] R. O. DUDA et P. E. HART : Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[13] J. ENS et P. LAWRENCE : An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):97 –108, 1993.

[14] O. FAUGERAS, B. HOTZ, H. MATHIEU, T. VIÉVILLE, Z. ZHANG, P. FUA, E. THÉRON, L. MOLL, G. BERRY, J. VUILLEMIN, P. BERTIN et C. PROY : Real time correlation-based stereo : algorithm, implementations and applications. Rapport technique RR-2013, INRIA, 1993.

[15] P. F. FELZENSZWALB et D. P. HUTTENLOCHERI : Efficient belief propagation for early vision. *The International Journal of Computer Vision*, 70(1):41–54, 2006.

[16] M. FIALA : ARTag revision 1. a fiducial marker system using digital techniques. Rapport technique NRC/ERB-1117, National Research Council of Canada, 2004.

[17] M. A. FISCHLER et R. C. BOLLES : Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] M. A. FISCHLER et R. C. BOLLES : A ransac-based approach to model fitting and its application to finding cylinders in range data. Dans *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 637–643, 1981.

[19] G. FORLANI, C. NARDINOCCHI, M. SCAIONI et P. ZINGARETTI : Building reconstruction and visualization from lidar data. Dans *Proceedings of International Archives of Photogrammetry and Remote Sensing*, vol. 34(5/W12), pages pp. 151–6, 2003.

[20] A. FUSIELLO, E. TRUCCO et A. VERRI : A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[21] D. G. HAKALA, R. C. HILLYARD, B. E. NOURSE et P. J. MALRAISON : Natural quadrics in mechanical design. Dans *Proceedings of the Autofact West*, pages 363–378, 1980.

[22] C. HARRIS et M. STEPHENS : A combined corner and edge detector. Dans *Proceedings of the* 4^{th} *Alvey Vision Conference*, pages 147–151, 1988.

[23] R. HARTLEY et A ZISSERMAN : *Multiple View Geometry in Computer Vision.* Cambridge University Press, New York, NY, USA, 2004.

[24] P. S. HECKBERT : Color Image Quantization for Frame Buffer Display. *SIGGRAPH Computer Graphics*, 16(3):297–307, 1982.

[25] H. HOPPE, T. DEROSE, T. DUCHAMP, J. MCDONALD et W. STUETZLE : Surface reconstruction from unorganized points. Dans *Proceedings of the* 19^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 71–78, 1992.

[26] P. V. C. HOUGH : Method and means for recognizing complex patterns. Brevet, 1962.

[27] Creaform INC. : Auto-referenced system and apparatus for three-dimensional scanning. Brevet, 2006.

[28] S. INOKUCHI, K. SATO et F. MATSUDA : Range imaging system for 3-d object recognition. Dans *Proceedings of the 7th International Conference on Pattern Recognition*, pages 806–808, 1984.

[29] F. ISGRÒ et E. TRUCCO : Projective rectification without epipolar geometry. Dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1094–1099, 1999.

[30] Vladimir KOLMOGOROV et Ramin ZABIH : Computing visual correspondence with occlusions using graph cuts. Dans *Proceedings of the 8th International Conference on Computer Vision*, vol. 2, pages 508–515, 2001.

[31] D. LÉVESQUE : Indices visuels de profondeur liés à la diffusion de la lumière. Mémoire de Maîtrise, Université de Sherbrooke, 2005.

[32] D. LOWE : Object recognition from local scale-invariant features. Dans *Proceedings of the International Conference on Computer Vision*, vol. 2, pages 1150–1157, 1999.

[33] G. LUKÁCS, R. MARTIN et D. MARSHALL : Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. Dans *Proceedings of the 5th European Conference on Computer Vision*, pages 671–686, 1998.

[34] Q.-T. LUONG et O.D. FAUGERAS : The fundamental matrix : theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17:43–75, 1995.

[35] J. B. MACQUEEN : Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pages 281–297. University of California Press, 1967.

[36] John MALLON et Paul F. WHELAN : Projective rectification from the fundamental matrix. *Image and Vision Computing*, 23:643–650, 2005.

[37] N. OTSU : A threshold selection method from gray-level histograms. IEEE *Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[38] M. POLLEFEYS, R. KOCH et L. VAN GOOL : A simple and efficient rectification method for general motion. Dans *Proceedings of the International Conference on Computer Vision*, pages 496–501, 1999.

[39] W. H. PRESS, S. A. TEUKOLSKY, W. T. VETTERLING et B. P. FLANNERY : *Numerical Recipes 3rd Edition : The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 3 édition, 2007.

[40] T. RABBANI et F. A. van den HEUVEL : Efficient hough transform for automatic detection of cylinders in point clouds. Dans *Proceedings of the International Society for Photogrammetry and Remote Sensing Workshop Laser scanning*, pages 60–65, 2005.

[41] M. ROUHANI et A. D. SAPPA : A novel approach to geometric fitting of implicit quadrics. Dans *Proceedings of the* 11^{th} *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 121–132, 2009.

[42] P. J. ROUSSEEUW et A. M. LEROY : *Robust regression and outlier detection.* John Wiley & Sons, Inc., New York, NY, USA, 1987.

[43] P. K. SAHOO, S. SOLTANI, A. K. C. WONG et Y. C. CHEN : A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41:233–260, 1988.

[44] J. SALVI, J. PAGES et J. BATLLE : Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.

[45] D. SCHARSTEIN et R. SZELISKI : A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *The International Journal of Computer Vision*, 47(7):131–140, 2002.

[46] R. SCHNABEL, R. WAHL et R. KLEIN : Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[47] M. SEZGIN et B. SANKUR : Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13:146–165, 2004.

[48] Richard SZELISKI, Ramin ZABIH, Daniel SCHARSTEIN, Olga VEKSLER, Vladimir KOLMOGOROV, Aseem AGARWALA, Marshall TAPPEN et Carsten ROTHER : A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1068–1080, 2008.

[49] Marshall F. TAPPEN et William T. FREEMAN : Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. Dans *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2, pages 900–907, 2003.

[50] F. TARSHA KURDI, T. LANDES et P. GRUSSENMEYER : Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. Dans *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences Workshop on Laser Scanning*, pages 407–412, 2007.

[51] R TSAI : A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3:323–344, 1987.

[52] G. VOSSELMAN : Building reconstruction using planar faces in very high density height data. Dans *Proceedings of International Archives of Photogrammetry and Remote Sensing*, pages 87–92, 1999.

[53] L. ZHANG, B. CURLESS et S. M. SEITZ : Spacetime stereo : Shape recovery for dynamic scenes. Dans *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 367–374, 2003.

[54] L. ZHANG et S. NAYAR : Projection defocus analysis for scene capture and image display. *ACM Transactions on Graphics*, 25:907–915, 2006.